A HEURISTIC PROGRAM FOR SOLVING BONGARD PROBLEMS

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By

MAHADEVAN SRIDHAR

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY, 1983

EE-1983-M-SRI-HEU

CERTIFICATE

This is to certify that the thesis entitled 'A HEURISTIC PROGRAM FOR SOLVING BONGARD PROBLEMS' by Mahadevan Sridhar has been carried out under our supervision and has not been submitted elsewhere for a degree.

15.7.83

(R.M.K. SINHA)
Assistant Professor
Dept. of Electrical Engg. and
Computer Science Programme
Indian Institute of Technology
Kanpur-208016

(P.R.K. RAO)
Professor
Dept. of Electrical Engg
Indian Institute of Technol
Kanpur-208016

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my advisors for undertaking to supervise this thesis. To Dr. P.R.K. Rao, for his keen and constructive criticism and willingness to learn new concepts, and to Dr. R.M.K. Sinha for giving me the benefit of his knowledge and experience in the field of Artificial Intelligence.

I must thank all the members of the Computer Science faculty for helping to create a convivial atmosphere for research. I am grateful to Rajeev Sangal for his consistent encouragement and for many discussions about AI. Thanks must go to Harish, for his never ceasing flow of ideas and for many valuable discussions covering this thesis.

Finally I would like to thank Mr. S.K. Tewari for his excellent typing.

M. Sridhar

(M. SRIDHAR)

ABSTRACT

This thesis is devoted to a study of the Bongard problem. A Bongard problem is composed of two classes of boxes, each box containing an arbitrary collection of 2-D figures. The solution to a problem consists in determining a rule distinguishing one class from the other. Rules are usually combinations of geometric features and spatial relations.

A methodology for solving Bongard problems is described whereby the process of rule discovery is likened to a best first search process in the space generated by the problem representation, guided by relevant heuristics. Evaluation functions are used to rate the nodes of the search space. The best first strategy is implemented in the program as an agenda control structure. Several sample solutions of Bongard problems as generated by the program are included.

It is claimed that this model for problem solving is powerful enough to solve the Bongard problems. Limitations of the methodology concern learning by automated addition of new heuristics.

CONTENTS

CHAPTER 1

INTRODUCTION

The ability of humans to detect 'patterns' from a given set of 'instances' is well known. This ability finds many uses, one of which is concerned with categorizing instances into mutually exclusive classes, where an instance belongs to a particular class if it satisfies the rule defining the class. A rule may also be interpreted as some distinguishing property of a class, which is shared by all its members. Conversely, given a rule the set of instances satisfying it, which form the class as a whole, are termed its positive instances or examples, while those not satisfying it are termed its negative instances.

The specific problem considered in this thesis can be stated more generally as

'Given both a set of positive instances of a rule and a set of negative instances of it, to discover the rule'.

Various methodologies or paradigms, fundamentally different in approach have been developed in an effort to solve the above problem. A brief discussion of some of these will be helpful in placing this work in relation to extant theories in the area.

1.1 Statistical pattern recognition

Most work in this area uses the following formulation of the problem - the set of positive instances is considered as constituting trial examples of one of a mutually exclusive set of prototypes. Then, given a test instance to be classified, the task is to determine the prototype to which it corresponds [NILSSON 65].

Briefly, the methodology used is as follows. Each prototype is described by a set of attributes and each attribute by a list of attribute values. The set of attribute values is then considered as constituting a space of appropriate dimension and each sample is represented as a point in the space. By assigning each prototype to a subspace of the attribute space, a partitioning of the latter is achieved. Instances are classified by their locations in the attribute space, determined by computing their attribute values. As the attributes are often considered to be statistically described, the original classification problem is then formulated in the framework of statistical decision theory. This approach has been used in many applications with varying degrees of success.

Significant limitations of this approach in solving many types of problems have been pointed out elsewhere [NARASIMHAN 69]. One criticism of this approach relevant

to the class of problems studied in this thesis is that by representing each instance as a point in the attribute space, information concerning the structure of the instance as relations between its parts is lost. Specifically consider the case when the rule concerns the spatial distribution of parts of the instance, like for example - the rule is that in all examples of a prototype, a triangle occurs inside of a circle. Any recognition system using statistical techniques will function poorly in this context since no clustering of prototypical examples in the attribute space can be expected. This is due to the difficulty in representing relations between parts of an instance by the attribute value representation scheme.

1.2 Rule discovery as search

A powerful paradigm in the area of problem solving views the process of rule discovery as a search in the space generated by the problem representation [NILSSON 80].

A reformulation of the rule discovery problem under the above paradigm is the following:

Given a set of instances P and a set of instances N, repeatedly generate a statement or predicate S about P and N, until every instance in P satisfies S and no instance in N satisfies S.

The rule, is that statement S which satisfies the above condition.

The issue now becomes one of evolving an efficient method by which the space can be searched for the rule. Various methods have been proposed to solve the above problem.

(i) A sophisticated search strategy: A straightforward approach is to devise an algorithm which, given a search space, is computationally the least expensive method of searching the space. Many search strategies have been designed for different kinds of problems [NILSSON 71]. However, since the size of the search space increases with the complexity of the problem, there are practical limitations of this approach for many problems.

(ii) A better problem representation: In some cases the size of the search space can be reduced by selecting an optimum representation for the problem. Thus even a simple search strategy would be sufficient to discover the rule in a reduced search space. Again, many representations have been evolved which are effective only for special types of problems.

(iii) Heuristic knowledge based search: Here the view point is that a problem solving program which possesses a lot of knowledge about the domain in which it operates has to do a

minimal amount of searching for the rule. The issue then becomes what kind of knowledge should be stored and how to store it.

Various knowledge representation schemes have been evolved. One such knowledge representation scheme consists in storing all the knowledge as 'thumb rules' or heuristics, usually represented as procedures in some programming code. Each heuristic rule is useful only in a certain number of situations. The issue then is to discover good heuristics which perform well over as wide a range as possible.

Lenat [LENAT 76] has convincingly demonstrated the power of good heuristics in systems which synthesise new rules or concepts. His system uses three kinds of knowledge, two of which are domain-independent.

(i) The first kind includes a lot of empirical rules for determining how interesting a concept is. For example· A concept is interesting if a lot of examples of it can be found.

(ii) The second kind guides the search for new concepts. A useful rule for doing this is: If little or few examples of a concept have been discovered try to generalise it.

(iii) Finally domain-dependent knowledge in the form of an initial base of concepts are included. Example domains are Set-theory and recently VLSI design and game playing [LENAT 83].

1.3 Some general features of the methodology used

The program described in this thesis possesses the following features characteristic of heuristic problem-solving programs.

(i) A corpus of heuristic rules: A number of such 'rules of thumb' are used by the program, which provide it with judgemental criteria for the task of generating new combinations of attributes and relations as possible rule candidates.

(ii) Evaluation functions: In order to enable comparison between the relative merits of two or more search paths, functions are provided which map rule possibilities and attributes into non-negative numbers.

(iii) Best-first search mechanism: The program maintains an agenda of possible paths in the search space from which the most promising one is selected and explored during each cycle of search.

An assumption implicit in the methodology described in the later chapters concerns the level of analysis of the problem. For example, a given visual scene will be represented in different ways for different purposes. Scene segmentation programs will operate on pixel representations whereas programs which do problem-solving would use a much richer symbolic representation of the scene. The representation used here is one of the latter kind, the assumption being that the problem of computing symbolic descriptions of the scene from the raw image and that of processing these symbolic representations to do problem-solving can be profitably studied separately.

1.4 Thesis outline

The following gives an overview of the thesis.

In chapter 2 we introduce the specific problem considered in this thesis. A representation for a typical problem is also described.

Chapter 3 provides motivation for the discussion on the methodology used to solve these problems. Specifically, the various components of the program are described briefly and an example solution of a problem as generated by the program is illustrated.

A detailed discussion of the methodology used is given in chapter 4 along with examples of heuristics and details of the search strategy.

Chapter 5 includes solutions to two more problems which illustrate many of the issues discussed in the previous chapter.

The last chapter concludes with a discussion of the capabilities and limitations of the methodology used.

Appendix 1 lists the set of heuristics used by the program. A list of sample problems is given in Appendix 2. Appendix 3 includes some details regarding the program implementation. Finally Appendix 4, gives an example of the application of our methodology to the problem of cell classification.

CHAPTER 2

THE PROBLEM

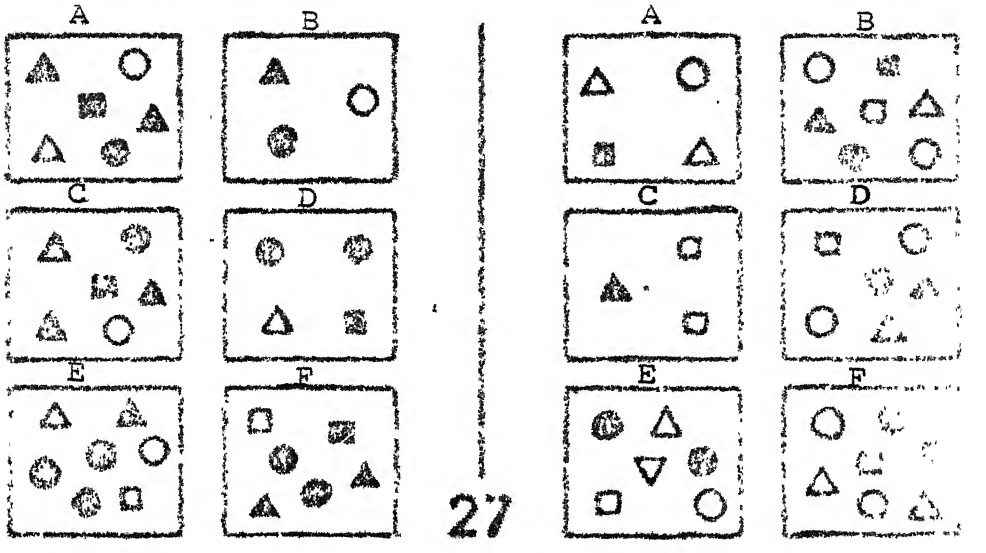
The first part of this chapter provides an introduction to the specific problem studied in this thesis, illustrated by examples. Following some remarks about the nature and scope of the problems considered, the chapter is concluded with a description of a model of a typical problem.

2.1 Statement of the problem

This thesis is concerned with the issues involved in the design of a program to solve the following kind of recognition problem.

A typical problem is depicted in figure 2.1. Each such problem is composed of twelve identical rectangular enclosures or boxes, divided into two equal sets or classes of six boxes each. Each box contains an arbitrary number of two-dimensional line drawings or figures. The order in which the boxes are placed in a class is immaterial. The following system of notation is used when referring to any particular problem. The class on the left hand side is called class I, that on the right hand side class II. In each class the boxes are labelled A to F, starting from the top left extreme box and proceeding from left to right in each row.

FIGURE 2.1



CLASS I

CLASS II

To solve a problem is equivalent to determining a rule or some distinguishing property which is unique to a class in the sense that it is true of every box belonging to the class and false of any box belonging to the other class. For example: The solution to the problem illustrated in figure 2.1 consists in determining the rule distinguishing any box in class I from that of any box in class II, namely 'In every box in class I, the number of 'shaded' figures is greater than that of 'outline' figures.'

Some remarks are in order concerning the nature of the problem statement. Firstly the number of boxes in a class is just a representative number (i.e. six) and clearly any number of boxes can be placed in a class. Secondly in the light of the previous chapter, the boxes in a class can be interpreted as constituting 'positive' instances of the rule defining the class. Though, the boxes of class II can be interpreted as 'negative' instances of the rule defined by class I, an important qualification which must be made is that in most of the problems considered here, class II also defines a precise rule. For example: In the case of the problem illustrated in figure 2.1, the rule defining class II is that in every box belonging to it the number of 'shaded' figures is less than the number of 'outline' figures. Hence there is a symmetry in each problem in the sense that each class constitutes negative examples of the other.

The problem in figure 2.1 is taken from a collection of one hundred such problems, which appeared in a book by the Russian scientist M. Bongard [BONGARD 70]. They have not merited much attention in the literature except for a brief analysis in [HOFSTADTER 79]. The number in boldface appearing in the centre of the problem in figure 2.1 indicates that it is the 27th one in the collection. We retain the original numbering for reasons of convenience. Henceforth we shall refer to these problems as the Bongard problems.

More examples of Bongard problems are illustrated in Appendix 2.

The computational solution of such problems has an 'open-ended' character since a large number of problems can be constructed. Indeed in the original collection there is a tremendous variation in the complexity of the problems, ranging from the almost trivial to some very subtle ones. However this open-endedness can be reduced in the following way :

By adopting a precise representation for individual figures and relations between them. This restricts the range of problem types to a well-defined set large enough to be interesting.

The methodology described in this thesis and its program implementation is based on the above representation. The claim is that there are very few problems in the original collection which are not solvable using this representation. Some justifications of the validity of the above claim is made in chapter 6. The limitations lie more towards the fact that learning to solve new problems by automatically synthesizing new heuristics from the existing set is not possible.

2.2 Modelling of a typical Bongard problem

We describe a representation for a typical Bongard problem in this section. This restricts the class of problems to those which can be expressed by the representation. We assume the existence of the following:

(i) A set F of attributes or features chosen from a larger set U of 2-D geometry descriptors. The elements of F would depend on the particular problem at hand. Some examples of features are: (a) shape (b) area (c) number-of-sides (d) axis (e) sketch etc.

(ii) Associated with each feature f(i) of the set F is a set A(i) of aspects or attribute values of the feature.

For example: The set {small, medium, large} could be one such aspect set associated with the feature area. Again the choice of aspects associated with a feature is flexible.

(iii) A set C of connectives using which features and aspects can be combined to yield composite features and composite aspects. A typical choice for C would be the set {and , or} which permits the disjunction and conjunction of features and aspects.

(iv) A set F^* comprising of composite features. If f(i) and f(j) are two features which are combined using the connective c(k), resulting in the composite feature f(i)-c(k)-f(j), the aspect set of this composite feature is computed as the set

{a(i)-c(k)-b(j) where for all i, a(i) ε A(i), the aspect set of f(i) and for all j, b(j) ε A(j), the aspect set associated with f(j)}

where a(i)-c(k)-b(j) denotes the combination of the aspects a(i) and b(j) by the connective c(k). For example: If f(i) is the feature area and f(j) is the feature number-of-sides, A(i) and A(j) being {small, medium, large} and {3,4}, c(k) being the connective 'and' , then the composite feature f(i)-c(k)-f(j) is area-and-number-of-sides whose aspect set is the union of the aspect sets:

{small-and-3, medium-and-3, large-and-3} and the set {small-and-4, medium-and-4, large-and-4} which intuitively refers to small, medium and large triangles and quadrilaterals.

(v) A set R_N of numeric relations and set-theoretic functions. Typical examples of this set are the relations greater-than, equal-to and less-than and the cardinality function.

(vi) A set R_S of spatial relations. These relations are typically anti-symmetric and transitive. Examples are the relations left-of, above-of and inside-of.

In terms of the above representation components of a typical problem i.e. figures, boxes and classes may now be interpreted as:

(i) A figure is represented as a list of feature aspect pairs. For example:

{(figure-type closed), (shape convex), (composed-of straight-lines),
(sketch outline), (number-of-sides 3), (axis vertical)}
would refer to a particular type of triangle.

(ii) A box is a collection of figures and a class is fixed collection of boxes (typically 6). A problem is comprised of two classes, each defining a rule. The

representation of these structures is considered in the next chapter.

We are now in a position to make the notion of a rule more precise. Let b be a box belonging to class c. Let S(b) be the set of true statements that can be asserted about the contents of b.A statement may be regarded as composed of symbols from the feature set F, the aspect sets A(i), the connectives set C, the composite feature set F^*, and the set R_N and R_S. Typically, statements regarding the contents of a box involve quantitative assertions about the number of figures possessing some combination of attributes and relations.

Given a class, we can associate with each box in that class an assertions set regarding the contents of that box. We define a conjecture about a class c to be a statement which belongs to the assertions set of each and every box in c, i.e. it is true of every box in the class c. Then, given a class c we can create and maintain a list of conjectures associated with it. Each conjecture is now a possible rule candidate.

A rule may be defined as that conjecture associated with a class, say class I, which is not true of any box in the opposite class.

To illustrate the above definitions with an example consider problem 27 in figure 2.1. The following statements can be interpreted as conjectures about the contents of class I.

(i) All figures are closed.

(ii) All figures are convex.

(iii) Every box contains a curvilinear figure.

(iv) Every box contains a shaded curvilinear figure.

(v) In every box, the number of shaded figures is greater than the number of outline figures.

Of all these conjectures only (v) survives the test for being a rule. All the others are true of at least one box in class II and hence cannot satisfy the prequisite for being a rule. It must be mentioned in passing that the rule distinguishing class I from class II and hence the solution to a problem may not be unique. So any conjecture which satisfies the above test will be a valid solution to a particular Bongard problem.

The rest of this thesis explicates this process of conjecture generation and testing which we propose as a model for Bongard problem solving.

CHAPTER 3

AN APPROACH TO SOLVING BONGARD PROBLEMS

We digress in this chapter to present a sample Bongard problem and its solution as worked out by the program. Thus an attempt is made to motivate the ensuing discussion on problem solving methodology by demonstrating that the theory advocated in the next chapter actually works in practice.

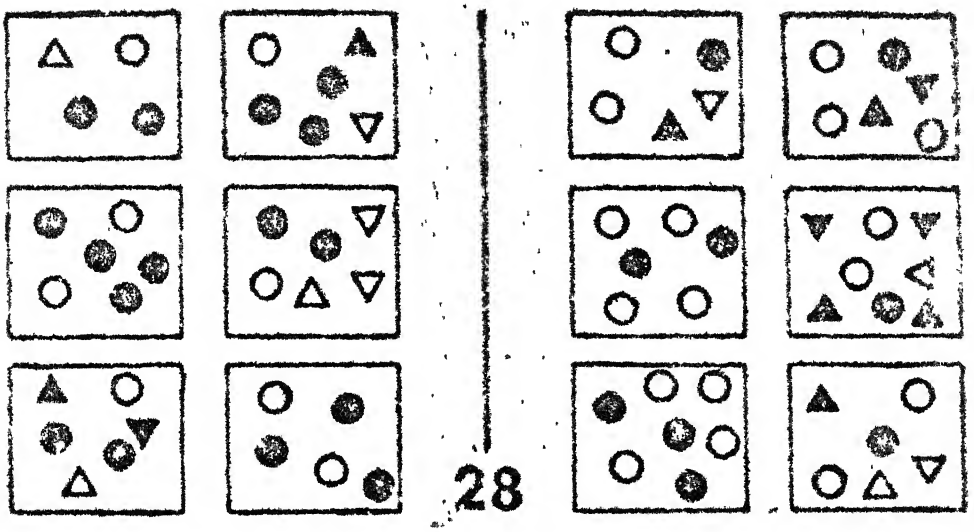
3.1 A sample problem

Consider problem 28 depicted in figure 3.1. The task is to demonstrate how the solution to this problem can be arrived at through a well defined computational process.

To this end, first we give a precise description of the representation chosen for this problem.

(i) The feature set F and associated aspect sets: The following features and their associated aspects are selected.

FIGURE 3.1



FEATURE	ASPECTS
Figure-type	(closed, open)
Composed-of	(straight-lines, curved-lines
Shape	(convex, concave)
Sketch	(outline , shaded)
Number of sides	(3,4,5)
Area	(small, medium, large)
Axis	(vertical, horizontal)

(ii) The set of connectives is the set (and, or).

(iii) Instead of listing out all the elements of the set of composite features F^* we define F^* as

$$F^* = \{f(k) \mid f(k) = f(i) - c(1) - f(j)\}$$

where the features $f(i)$ and $f(j)$ are either basic features or are themselves composite.

(iv) We take the numeric relations greater-than, equal-to and less-than as well as the set theoretic notions,member, cardinality and elements-of as the elements of the set R_N.

(v) To simplify matters, we assume that the set of spatial relations R_S is empty. Later we provide examples of solutions where this restriction is removed.

3.2 The program

A brief description of the various facets of the program will be helpful in understanding the solved example. In the next chapter a detailed analysis of these program components is given.

REPRESENTATION OF BOXES AND CLASSES

To solve a Bongard problem, the program must possess knowledge as to what kind of figures are present, how they are related and so on. Figures are represented, as mentioned earlier, by a list of feature aspect pairs. A straightforward method is to represent boxes as a list of such figure representations and classes as a list of such box representations. This scheme is not particularly suited for the kind of computations that need to be performed during the process of problem solving.

To answer questions of the form:

(i) which figures in this box are convex?

(ii) which boxes contain shaded figures? efficiently, clearly an alternative representation scheme is desired.

Figure 3.2 depicts the representation for a box, specifically box A in class I, problem 28. A box is represented by a unit called a structure. Each structure

Figure 3.2

structure AIP28

AXIS :

VERTICAL : (#1AIP28)

#-OF-SIDES :

THREE : (#1AIP28)

AREA :

SMALL : (#4AIP28 #3AIP28 #2AIP28 #1AIP28)

SKETCH :

SHADED : (#4AIP28 #3AIP28)

OUTLINE : (#2AIP28 #1AIP28)

SHAPE :

CONVEX : (#4AIP28 #3AIP28 #2AIP28 #1AIP28)

COMPOSED-OF :

CURVED-LINES: (#4AIP28 #3AIP28 #2AIP28)

ST-LINES : (#1AIP28)

MEMBER : IP28

FIG-TYPE :

CLOSED : (#4AIP28 #3AIP28 #2AIP28 #1AIP28)

ELEMENTS : (#1AIP28 #2AIP28 #3AIP28 #4AIP28)

ISA : BOX

Note: AIP28 refers to box A, class I in problem 28
#1AIP28 refers to figure 1 in the above box
IP28 refers to class I, problem 28.

is a collection of slots. A slot may represent many things. In the simplest case, a slot represents a feature, and the contents or value of the slot is an aspect associated with it. So a figure can be represented as a structure, whose slots are features and slot values are aspects associated with the features.

A slot itself may comprise of a collection of sub-slots. Such a slot has the following structure

Slot - 1 :

slot-1-1 : (contents of slot-1-1)
slot-1-2 : (contents of slot-1-2)
.
.
.
.
slot-1-n : (contents of slot-1-n)

Each sub-slot has the same structure as a slot. Consider, as an example, the slot representing the feature composed-of in figure 3.2. The composed-of slot is a collection of two sub-slots namely curved-lines and straight-lines. Each sub-slot in this case has just a value, which is the set of figures in the box which are composed of curved-lines and straight-lines respectively.

The structure for box A makes explicit the following information about the contents of A.

(i) A contains four figures.

(ii) A belongs to class I.

(iii) A contains small figures.

(iv) A contains shaded figures.

Figure 3.3 depicts a portion of the representation for a class as a structure, specifically class I in problem 28. One added level of complexity here is that each sub-slot is itself a slot in the sense of being composed of other sub-slots.

Consider as an example, the slot representing the feature sketch. The slot for sketch has two sub-slots, one representing the aspect shaded and the other representing the aspect outline. The sub-slot for outline is composed of six ternary slots, each of which represents a box in class I. These ternary slots have as their slot contents, the set of figures which are outline in the box represented by that slot.

One notational point to be made here is that if all the figures in a box possess the same aspect of a feature, the ternary slot in the class structure representing the box has the slot value T. Thus in figure 3.3 since all figures in box F, class I, are small, the ternary slot representing box F in the area slot has T as its slot value.

Figure 3.3

```
structure   IP28

AXIS :

   VERTICAL :
        EIP28 : (#6EIP28 #4EIP28 #1EIP28)
        DIP28 : (#6DIP28 #5DIP28 #2DIP28)
        BIP28 : (#6BIP28 #2BIP28)
        AIP28 : (#1AIP28)

#-OF-SIDES :

THREE :
      EIP28 : (#6EIP28 #4EIP28 #1EIP28)
      DIP28 : (#6DIP28 #5DIP28 #2DIP28)
      BIP28 : (#6BIP28 #2BIP28)
      AIP28 : (#1AIP28)

AREA :

   SMALL :
     FIP28 : T
     EIP28 : T
     DIP28 : T
     CIP28 : T
     BIP28 : T
     AIP28 : T

SKETCH :

   OUTLINE :
        FIP28 : (#4FIP28 #1FIP28)
        EIP28 : (#6EIP28 #2EIP28)
        DIP28 : (#6DIP28 #5DIP28 #4DIP28 #2DIP28)
        CIP28 : (#5CIP28 #2CIP28)
        BIP28 : (#6BIP28 #1BIP28)
        AIP28 : (#2AIP28 #1AIP28)

   SHADED :
        FIP28 : (#5FIP28 #3FIP28 #2FIP28)
        EIP28 : (#5EIP28 #4EIP28 #3EIP28 #1EIP28)
        DIP28 : (#3DIP28 #1DIP28)
        CIP28 : (#6CIP28 #4CIP28 #3CIP28 #1CIP28)
        BIP28 : (#5BIP28 #4BIP28 #3BIP28 #2BIP28)
        AIP28 : (#4AIP28 #3AIP28)

SHAPE :

   CONVEX :
      FIP28 : T
      EIP28 : T
      DIP28 : T
      CIP28 : T
      BIP28 : T
      AIP28 : T
```

other slots

The class I structure reveals the following information with minimum computation.

(i) Box C contains no straight-line figure, hence in class I every box does not contain straight-line figures.

(ii) Class I contains only small figures and so on.

Frequently the contents of a particular slot represent examples of an aspect of some feature. As we shall see shortly this is of central importance to the process of rule generation since most of the computation done by the program concerns the inspection of slot contents of boxes and classes and making appropriate conjectures on the basis of detected patterns.

The knowledge representation scheme outlined above was one of a declarative type, in the sense that the stored facts could be inspected, added to or modified.

In the first chapter, it was noted that heuristics were most often stored as procedures, in our case executable chunks of LISP code. We now turn to a description of this procedural type of knowledge representation, after outlining the search strategy used by the program.

3.3 The control structure

The process by which the program generates the rule solving a Bongard problem is primarily one of best-first search guided by relevant heuristic rules. Consider the two class structures, and their associated slots. Given a particular problem, the slots of the class structure are fixed by the set of features, aspects and relations chosen for the problem. The set of possible slot contents under this representation defines a space. By undertaking a best-first search of this space, the program discovers the rule essentially by examining the contents of slots representing the aspect distribution of features over the boxes of a class, creating new class slots for storing composite features and their examples and so on.

The best-first search strategy is implemented in the program as an agenda control structure. That is, at any given moment during the problem solving process, a list of plausible tasks is maintained by the program representing promising paths in the search space to finding the rule. The essential computation performed by the program is to repeatedly select one task from the agenda and execute it. A task may check a feature for the purpose of determining patterns involving its examples, suggest combining two or more features into a composite feature or add more tasks to the agenda.

A task on the agenda typically looks like the following: ''Check the conjecture - Every box in class I contains a cuvilinear figure in problem 28''. The selection of the most promising task from the agenda is facilitated by assigning to each task on the agenda a numeric rating. The task selected is the one with the highest worth. The details of the process by which tasks are rated is given in the next chapter. For the present it is sufficient to remember that the numeric rating of a task depends on the nature of the task, i.e. whether the task is a conjecture to be verified or a check of some sort and it depends on the rating of the features which appear as arguments to the task. Features are rated by a certain evaluation function which gives an approximate idea of its relevance to the rule at hand. The exact functions are given in the next chapter.

This agenda mechanism of best-first search is similar to that used in the AM program of Lenat [LENAT 76]. The only difference is that our program is goal-directed, while AM which is dedicated to the task of defining new interesting concepts from old ones, has no explicit goals to reach.

3.4 Heuristic rules

We have not yet indicated now tasks are going to be carried out and how class structures are expanded by adding more slots representing composite features. A task is usually represented as a executable chunk of LISP code and to carry out a task is to run this code chunk.

In our case, tasks are usually a series of LISP functions which are executed sequentially. The knowledge that is needed to recognise simple patterns or regularities in feature examples or even random occurences is encoded as a set of heuristic rules.

Carrying out particular tasks may require that knowledge needed for this purpose be somehow stored and retrieved while executing it. Knowledge particular to a task is represented as heuristics relevant to it in some way. That is, either the heuristics are encoded into the task rigidly or more flexibly a link is placed between a task and that heuristic which ensures that the heuristic will be accessed and used while carrying out the task.

In the above sense, heuristics are procedural representations of particular problem solving knowledge, which in our program is an executable piece of LISP code.

To illustrate the above ideas with an example, consider the following task: ''Specialise the aspect straight-lines of the feature composed-of in class I''

By specialising an aspect of a feature is meant creating a new composite aspect consisting of the conjunction of this aspect with another aspect (of another feature). For eg. The composite aspect straight-lines-and-shaded is a specialisation of the aspects straight-lines and shaded of the features composed-of and sketch respectively. Here since straight-line figures may be outline or shaded drawings, the aspect straight-lines-and-shaded is a specialisation of the aspect straight-lines.

One heuristic relevant to the above task is given below:

For aspect a(1) of feature f(1) and aspect a(2) of feature f(2),
IF (i) intersection of the aspects a(1) and a(2) is non-null
(ii) all examples of a(1) are not examples of a(2)
(iii) all examples of a(2) are not examples of a(1)
(iv) worth (f(1))/worth (f(2)) < 4.0
THEN

a(1)-and-a(2) is an interesting aspect composition, so add the following task to the agenda: Fill-in examples of a(1)-and-a(2) whose worth is computed as average (worth (f(1)), worth (f(2))).

If for example, in a particular Bongard problem, the only straight-line figures are triangles then the composite aspect three-sided-and-straight-lines will not be formed as it does not satisfy condition (ii) and (iii) of the IF part of the above heuristic.

3.5 The solution to the problem

We conclude this brief description of the program components and now turn to the example solution. This consists of a trace of the program execution showing the chronological order in which tasks were suggested and carried out. This will illustrate some of the points discussed above.

Some explanation is due before presenting the actual example. The trace consists of a description of the tasks which were selected and the sequence in which they were carried out. An illustration of the problem (i.e. 28) has already been given in figure 3.1. The representation of boxes and classes as structures has been described earlier.

Each cycle signifies the selection and execution of a task from the agenda. The task description indicates the nature of computation performed by it. After the trace an

analysis of the solution is presented, where the stages of progress towards the rule are identified.

Chapter 5 provides two more examples of solutions generated by the program, which will further illustrate the methodology used to solve these problems.

As shown in the trace, when a task has been selected a description of it is printed out to the user. These descriptions are opaque pieces of English text.

3.6 SOLUTION TRACE OF PROBLEM 28

CYCLE # 0

1 TASK(S) ON AGENDA

Initialising the list (FIGURE-TYPE COMPOSED-OF SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) of individual features for investigation in class IP28

CYCLE # 1

1 TASK(S) ON AGENDA

Investigating the feature FIGURE-TYPE for patterns in its examples in class IP28

CYCLE # 2

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP28 is CLOSED

...testing the conjecture on the boxes of class IIP28

conjecture is an invalid rule candidate for class IP28

since it is true of the boxes (FIIP28 EIIP28 DIIP28 CIIP28 BIIP28 AIIP28) in class IIP28

this feature is being eliminated from the list of features relevant to the rule

---

CYCLE # 3

1 TASK(S) ON AGENDA

Investigating the feature SHAPE for patterns in its examples in class IP28

CYCLE # 4

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP28 is CONVEX

...testing the conjecture on the boxes of class IIP28

conjecture is an invalid rule candidate for class IP28

since it is true of the boxes (FIIP28 EIIP28 DIIP28 CIIP28

BIIP28 AIIP28) in class IIP28

this feature is being eliminated from the list of features

relevant to the rule

CYCLE # 5

1 TASK(S) ON AGENDA

Investigating the feature AREA for patterns in its examples

in class IP28

CYCLE # 6

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP28 is SMALL

...testing the conjecture on the boxes of class IIP28

conjecture is an invalid rule candidate for class IP28

since it is true of the boxes (FIIP28 EIIP28 DIIP28 CIIP28

BIIP28 AIIP28) in class IIP28

this feature is being eliminated from the list of features

relevant to the rule

CYCLE # 7

1 TASK(S) ON AGENDA

Investigating the feature COMPOSED-OF for patterns in its examples in class IP28

CYCLE # 8

2 TASK(S) ON AGENDA

Conjecture : There is a CURVILINEAR figure in every box in class IP28

Testing it on the boxes of class IIP28

...conjecture is an invalid rule candidate for class IP28 since it is true of boxes (FIIP28 EIIP28 DIIP28 CIIP28 BIIP28 AIIP28) in class IIP28

CYCLE # 9

2 TASK(S) ON AGENDA

Specialise the aspect CURVILINEAR of the feature COMPOSED-OF in class IP28

CYCLE # 10

4 TASK(S) ON AGENDA

Filling in examples of the intersection of CURVILINEAR figures and OUTLINE figures in class IP28

CYCLE # 11

4 TASK(S) ON AGENDA

Filling in examples of the intersection of CURVILINEAR figures and SHADED figures in class IP28

CYCLE # 12

3 TASK(S) ON AGENDA

Investigating the feature SKETCH for patterns in its examples in class IP28

CYCLE # 13

5 TASK(S) ON AGENDA

Conjecture : There is a OUTLINE figure in every box in class IP

Testing it on the boxes of class IIP28

...conjecture is an invalid rule candidate for class IP28

since it is true of boxes (FIIP28 EIIP28 DIIP28 CIIP28 BIIP28 AIIP28) in class IIP28

CYCLE # 14

5 TASK(S) ON AGENDA

Conjecture : There is a SHADED figure in every box in class IP2

Testing it on the boxes of class IIP28

...conjecture is an invalid rule candidate for class IP28

since it is true of boxes (FIIP28 EIIP28 DIIP28 CIIP28 BIIP28 AIIP28) in class IIP28

CYCLE # 15

5 TASK(S) ON AGENDA

Specialise the aspect OUTLINE of the feature SKETCH in class IP2

CYCLE # 16

6 TASK(S) ON AGENDA

Specialise the aspect SHADED of the feature SKETCH in class IP2

CYCLE # 17

7 TASK(S) ON AGENDA

Filling in examples of the intersection of OUTLINE figures and
STRAIGHT-LINE figures in class IP28

CYCLE # 18

7 TASK(S) ON AGENDA

Filling in examples of the intersection of SHADED figures and
STRAIGHT-LINE figures in class IP28

CYCLE # 19

6 TASK(S) ON AGENDA

Checking the feature COMPOSED-OF-AND-SKETCH for patterns
involving its examples in class IP28

CYCLE # 20

8 TASK(S) ON AGENDA

Conjecture : There is a CURVILINEAR-AND-SHADED figure in every box in class IP28

Testing it on the boxes of class IIP28

...conjecture is an invalid rule candidate for class IP28 since it is true of boxes (FIIP28 EIIP28 DIIP28 CIIP28 BIIP28 AIIP28) in class IIP28

CYCLE # 21

8 TASK(S) ON AGENDA

Conjecture : There is a CURVILINEAR-AND-OUTLINE figure in every box in class IP28

Testing it on the boxes of class IIP28

...conjecture is an invalid rule candidate for class IP28 since it is true of boxes (FIIP28 EIIP28 DIIP28 CIIP28 BIIP28 AIIP28) in class IIP28

CYCLE # 22

8 TASK(S) ON AGENDA

Conjecture : In every box in class IP28 examples of CURVILINEAR-AND-SHADED figures are GREATER-THAN examples of CURVILINEAR-AND-OUTLINE figures

Testing it on the boxes in class IIP28

Conjecture is a valid rule candidate for class IP28 !!!

TASK : STATE-RULE (IP28 COMPOSED-OF-AND-SKETCH CURVILINEAR-AND-SHADED GREATER-THAN CURVILINEAR-AND-OUTLINE))

RULE : CLASS IP28 FEATURE COMPOSED-OF-AND-SKETCH HAS-VALUE (CURVILINEAR-AND-SHADED GREATER-THAN CURVILINEAR-AND-OUTLINE).

3.7 ANALYSIS OF THE SOLUTION TO PROBLEM 28

A study of the solution to problem 28 as generated by the program is undertaken here. The various stages of the problem-solving process are identified as groups of tasks accomplishing particular objectives.

STAGE 1 : Cycles 0 - 5

Three of the basic features were analysed in this stage. Three conjectures, identical in form, were proposed as rule candidates all of which turned out to be invalid. Then the three features were eliminated from further consideration as their irrelevance to the rule becomes firmly established.

STAGE 2 : Cycles 7 - 11

The feature composed-of was investigated next by studying its examples in class I. Definite progress towards the solution is made for the first time as the aspect 'curvilinear' of the feature composed-of is specialised after the latter generates an invalid conjecture. Examples of curvilinear shaded figures and curvilinear outline figures are filled in as slot contents of the slot representing the newly created composite feature composed-of-and-sketch in the class I structure.

STAGE 3 : Cycles 12 - 18

The feature sketch was studied at this stage, which generated another invalid conjecture. Then the program temporarily veers off and fills in examples of shaded straight-line figures and outline straight-line figures. This endeavour is useless as the solution to the problem demonstrates. However since the examples of the above two composite aspects occur randomly in class I, they are ignored henceforth.

STAGE 3 : Cycles 19 - 22

The crucial cycle is the 19th one where the feature composed-of-and-sketch was investigated for patterns involving its examples in class I. Three conjectures were proposed, two of which turn out to be invalid. Finally, the third one yields the desired rule differentiating class I from II.

CHAPTER 4

A METHODOLOGY FOR SOLVING BONGARD PROBLEMS

In this chapter a methodology for solving Bongard problems is presented. The various facets of the program namely heuristics, evaluation functions and the agenda control structure are described in more detail, along with techniques to represent relations and discover rules involving them. In the next chapter some examples of Bongard problems solved by such an approach are presented.

First the basic theory is presented in three stages. Initially we assume that a rule must involve individual features. The next section extends this simple case to rules involving combinations of two or more features. Finally we consider the case when spatial relations are allowed as rule constituents.

Next, the need for evaluation functions is discussed along with the functions themselves. Finally some details are included about the agenda mechanism that were briefly mentioned in the previous chapter.

4.1 Aspect distribution of features

A feature can be regarded as a basic ingredient of a rule since the latter is composed of various combinations

of the former. Then, it is of crucial importance to develop some method whereby the relevance of a feature to a rule can be ascertained by some simple computation.

One parameter which is useful in this regard is the distribution of aspects of a feature over the boxes of a class. By this we mean the following: (i) How many aspects of the feature actually occur in a class?
(ii) In what way does the feature occur in the figures of a box?

This information is important since on this basis assertions can be made about box and class contents which lead to conjectures about classes and finally the rule itself.

Specifically the following categories of feature occurences suggest themselves.

(i) Features can assume one or more aspects over the boxes in a class. For example: A class may contain straight-line figures and curvilinear ones. In this case the feature composed-of has the above two aspects in this class.

(ii) Features can occur in some or all the boxes in a class. This implies that it is possible that no figure in a certain box is describable by a feature. For example: If a box contains only curvilinear figures, the feature number-of-sides will be absent from this box.

(iii) Features can occur in some or all the figures in a box. That is, in a box in which a feature occurs, it may be used to describe some or all the figures in that box.

The above general considerations can be used to formulate the following heuristic rules for investigating the occurences of aspects of any feature.

HEURISTIC 1:

For a feature f,

IF number of aspects of f = 1
and f occurs in every box b belonging to class c
and every figure belonging to b is describable by f
THEN add the following task to the agenda -
'Conjecture: Every figure in c has aspect a of feature f
whose worth is computed as 2 x worth (f).

This straightforward heuristic is useful in two ways. First if the conjecture added to the agenda turns out to be a valid rule candidate, the problem is solved. More frequently this heuristic is useful in eliminating features, in some sense trivial, from further consideration thus pruning the search space considerably. For example: In problem 4 (see appendix 2)the suggested conjecture directly yields the rule. In contrast, consider problem 28 (figure 3.1).

The feature area satisfies the IF part of this heuristic and so the conjecture 'All figures in class I, problem 28, are small in area' is added to the agenda. When this task is picked up and executed, since class II also contains small figures this conjecture is an invalid rule candidate. Hence the feature area is clearly not relevant to the rule being sought and is eliminated. This process of feature elimination is useful in containing the combinatorial explosion that will result when a large number of features are needed to describe pattern samples. How this process of feature elimination is actually implemented will be explained in a later section. The worth or rating of the conjecture being added to the agenda is calculated as twice the worth of the feature. This is done to give the conjecture a high priority.

A straightforward generalisation of the above heuristic yields:

HEURISTIC 2:

For any aspect a of feature f in class c
IF f does not satisfy heuristic 1
and aspect a occurs in every box in c
THEN add the following task to the agenda -
'Conjecture: Every box b in class c has a figure
described by aspect a of feature f'
whose worth is computed as worth (f).

Since any f satisfying heuristic 1 automatically satisfies the second clause of the IF statement, we check explicitly for this. The above heuristic is useful in the same way as the first one is. For example: consider problem 28 depicted in figure 3.1. Every box in class I contains a curvilinear figure and the above heuristic 'triggers'. But the resulting conjecture is invalid as a rule for class I since the same is true for class II also. Obviously the feature composed-of cannot be eliminated from further consideration because of this single reason. A more prudent strategy is to label the aspect curved-lines of the feature composed-of as a candidate for feature combination heuristics to work on. This is done and as it turns out from the solution to problem 28, the combinations of the aspect curved-lines, namely curved-lines-and-outline and curved-lines-and-shaded, of the feature composed-of does yield the rule.

More frequently than not features will have multiple aspects in a class. This is certainly true for any fairly recondite rule differentiating class I from class II. One simple strategy is to run the above heuristic on each aspect of the feature. That is

For any feature f

IF number of aspects of f is > 1

THEN for each aspect a of f, execute heuristic 2.

This mini-heuristic turns out to be quite useful. Again consider the example of problem 28. The feature sketch has two aspects shaded and outline in class I. Hence heuristic 2 is run on both the aspects shaded and outline. Both computations yield conjectures which are invalid as rule candidates. Thus both the aspects are investigated by feature combination heuristics. This strategy again pays off, as is evident from the solution to problem 28.

A more interesting way to tackle features having multiple aspects is to discover numeric relations between the examples of the various aspects of the feature in each box. Specifically,

For any feature f

IF f has multiple aspects in class c

THEN for any aspect pair < a(1), a(2) >

belonging to the feature f,

if in every box in c, the examples of a(1) are related to the examples of a(2) by the numeric relation R, then suggest the following conjectures: 'In all boxes of class c, a(1) R a(2).'

This heuristic is useful in problems 27 (see appendix 2) and 28. Unfortunately the above heuristic is not very

useful in the case when the conjecture suggested by it fails as rule candidate. In this sense its use is somewhat more limited than the previous heuristics.

Consider the case when the feature f occurs more randomly in class c. Specifically, when f does not occur in all the boxes of c, can f be ignored? No (see problem 13 in appendix 2 for a counter example), since it may turn out that some combination of f with another feature f' could possibly result in a more interesting composite feature. In the next section we shall describe a heuristic which looks for this sort of 'random' character in a feature.

Before concluding this section the following clarifications must be pointed out.

(i) This list of heuristics is only a sample of the set of possible ones. Many more are used by the program to investigate individual features. To take one example, consider the following heuristic:

For any feature f

IF f has multiple aspects in c
and f has a single aspect a over every box in c
THEN suggest the following conjecture: 'The feature f is single valued over the boxes of c.'

This heuristic is useful in problems 22 and 56 (see appendix 2).

(ii) The above heuristics have not been written with specific problems in mind, as it might appear from the examples we have illustrated. Each heuristic is useful in a variety of problems and also in a variety of ways. For example:Heuristic 1 is useful in problems 2,3,4,5,6 and 23. Heuristic 2 is useful in problems 21, 24, 25 and 26 etc.

(iii) The feature investigated by these heuristics may be a basic feature or it might be a composite one formed from two or more basic ones. For example: The heuristic which determines numerical relationships between the aspects of a feature was successful in problem 27 on sketch which is a basic feature but the same heuristic worked in problem 28 on sketch-and-composed-of to discover the rule which is a composite feature. The implication of this is that these strategies are not 'once' useful types but every time a new feature is created out of the existing features, these heuristics are used to investigate them.

(iv) Finally, it is not necessary that these heuristics are useful only when they suggest conjectures which turn out to be valid rules. More frequently when a conjecture which is suggested by one of these heuristics fails, some valuable information has been obtained about the behaviour of the feature occurence over the class. To give two examples of this:

(a) When the feature is discovered to be irrelevant to the rule, as outlined earlier feature elimination processes are activated which result in a pruning of the search space.

(b) Features which do not by themselves yield the rule are sometimes passed over to feature combination heuristics for study. Problems 13 and 28, to take but two examples, provide graphic illustrations of the importance of such a strategy.

4.2 Combinations of features

In the previous section some heuristics were described for investigating patterns involving the aspect examples of individual features. It was emphasized that cases in which conjectures suggested by those heuristics turned out to be invalid rule candidates are important for the purpose of guiding further search.

Here we illustrate the above claim by describing heuristics which form combinations of two or more features, usually by intersections and unions of the set of examples of the features being combined. In a sense, this section is concerned with the second pass of the feature investigation process. In the first pass, study of the basic features usually results in either the solution to the problem or in the elimination of some basic features from further

consideration. The second stage involves executing tasks dedicated to forming new feature combinations and their study. The process is then repeated by adding the list of newly created features to the existing list of relevant features to the solution of the problem at hand.

To take an example, assume the task 'Conjecture: Every box in class I contains a curvilinear figure' has been selected from the agenda. The conjecture is checked by determining whether class II also shares this property. If it does, then this conjecture is obviously an invalid rule candidate. This causes a new task to be added to the agenda which searches for interesting combinations of features involving the feature composed-of and it's aspect curved-lines. A heuristic useful to the above task is given below.

For checking the conjecture that every box in class c
has aspect a of feature f,
IF a does not occur in any box in the class - c
THEN state the following rule: Every box in class c
has aspect a of feature f.
OTHERWISE add the following task to the agenda:
Specialise aspect a of feature f in class c,
whose worth is computed as worth (f).

Here if class c is I then -c is the class II.

By specialising the aspect a of feature f is meant selecting some other feature f' and forming the composite feature f-and-f' whose aspect set is computed as defined in the second chapter. Not all the aspect combinations are computed. Specifically, only those are created which satisfy some requirements. One heuristic useful in determining whether an aspect conjunction is a useful one is the following:

For any aspect a(1) of feature f(1)
and aspect a(2) of feature f(2), in class c
IF (i) f(1) and f(2) have non-zero worths
(ii) worth (f(1))/worth (f(2)) < 3
(iii) the intersection of examples of a(1) and a(2) is non-null
(iv) all examples of a(1) are not examples of a(2)
(v) all examples of a(2) are not examples of a(1)
THEN add the following task to the agenda:
Fill-in intersection of the aspects a(1) and a(2)
of the features f(1) and f(2) respectively in class c
whose worth is computed as average (worth (f(1)), worth (f(2))).

A fill-in task causes a new attribute/slot to be added to the class structure. In the above case, examples of the composite aspect a(1)-and-a(2) will be stored under the slot f(1)-and-f(2).

As an example of the use of the above heuristic, consider the example of problem 28. Suppose the following task has been selected from the agenda: Specialise the aspect curved-lines of the feature composed-of in class I. Executing the above task implies that the feature composed-of is combined with each one of the existing set of features and the pair is studied by the above heuristic. The precise manner in which this iterative process is incorporated into the agenda scheme is the topic of the final section of this chapter. For the above example execution of this heuristic yields in the following task being added to the agenda, among others.

'Fill-in examples of the intersection of curvilinear figures and outline figures in class I'.

Let us examine the reasons for this choice: (a) worth (composed-of) = 30 which is > 0. Similarly, worth (sketch) = 30 > 0. Hence condition (i) of the above heuristic is satisfied. (b) worth(composed-of)/ worth (sketch) = 1 < 3 so (ii) is satisfied. (c) Since examples of outline curvilinear figures can be found in class I (iii) is satisfied. (d) Finally, (iv) and (v) are satisfied since there exist curvilinear figures which are not outline and outline figures which are not curvilinear.

We now turn to another important method for combining features, namely by feature generalisation. In the previous section on single feature analysis, we remarked that frequently features having random occurence, in the sense of not being present in all boxes in a class, are found. We examine one way to discover patterns involving examples of such features.

Consider the following heuristic :

. For any feature f in class c,
IF f does not occur in every box in c,
THEN add the following task to the agenda:
Form compositions of features involving aspect a
of feature f in c
whose worth is computed as worth (f).

One way to form composite features is by conjunction as we illustrated earlier. Another way is to form disjunctions of two or more features. The heuristic for this is parallel to the one for forming conjunctions except that in this case unions of aspect examples are formed and not intersections.

To see an example of where this heuristic may be useful, consider problem 13 in appendix 2. The feature number-of-sides and its aspect 'four' satisfy the above heuristic and in the search for generalisations of the feature number-of-sides, the following task is suggested:

Fill-in examples of the union of four-sided figures and curvilinear figures in class I.

The resulting aspect which is four-or-curvilinear of the feature number-of-sides-or-composed-of occurs in all the boxes and hence here we have a case where a feature having random occurences in a class forms an interesting combination. However, even this composite feature does not yield the rule and as is clear from problem 13 a combination of the basic features number-of-sides, axis and composed-of finally yields the rule. The complete trace of the solution to this problem is given in the next chapter.

We conclude this section on feature combinations by summarising the salient points discussed so far:

(i) Two important methods for combining features are by specialisation and generalisation.

(ii) These techniques can be used to form combinations of two or more features. The features being combined may be basic features or composite ones.

(iii) The combination of two or more features has again the same structure as that of a basic feature. Hence the heuristics for investigating individual features can be used to study these composite ones also.

4.3 Representing relations and discovering rules involving them

So far, we have not considered how spatial relations between figures in a box could be represented and what kind of knowledge is needed to discover rules involving them. Here we describe an extension to the basic methodology expounded in the previous two sections by including spatial relations in the repertoire of rule constituents.

Relations between figures in a box are represented using the slot/contents technique used to represent features and their associated aspects. Specifically, all facts concerning the spatial relationships of figures in a box is recorded by appropriate slots and their contents in the structure representing the box. A typical representation of this information is depicted in figure 4.1. For example: The attribute left-of of the slot spatial-relations lists pairs of figures belonging to box A such that the first figure of the pair is left-of the second figure.

One important difference between features and relations concerns the suitability of the initial representation towards discovering patterns corresponding to conjectures about class contents. For the case of features the initial representation of features as slots and aspects as attributes was sufficient to compute rules involving them. By contrast the above method for representing relations is not directly suitable for the purpose of computing rules involving them.

For example, consider problem 37 (see appendix 2). Here in each box of a class, spatial information is recorded in the form, figure x is related to figure y by a relation R. Using this information one cannot directly deduce that, to take an example, in boxes A and F in class I a four sided figure is left of a curvilinear figure. Since spatial information is stored as relations between figures represented by names, first the relevant properties of the figures must be retrieved before any box independent statement can be asserted about the spatial relation.

This representation problem is solved in the following manner. Under the spatial-relation slot in each box is stored information concerning the spatial distribution of figures in the box, as outlined earlier. However, in the class structure this spatial information is represented in a different manner.

Consider the example of class I, problem 37. In the class I structure facts concerning the spatial distribution of features is stored as shown in figure 4.2, where only the relevant slots have been depicted. The meaning of the aspect 3-left-of-composed-of of the slot #-of-sides-left-of-composed-of is that it represents all figure pairs such that the first figure has three sides and the second figure is a curvilinear figure. It is clear that this kind of

FIGURE 4.1

```
Structure  AIP37
#-OF-SIDES :
     4 : (# 3AIP37)
     3 : (# 1AIP37)
AREA :
   SMALL : (# 3AIP37 # 2AIP37 # 1AIP37)
SKETCH :
   OUTLINE : (# 3AIP37  # 2AIP37  #1AIP37)
SHAPE :
   CONVEX :  (# 3AIP37  # 2AIP37  # 1AIP37)
COMPOSED-OF :
   CURVED-LINES : ( # 2AIP37)
   STRAIGHT-LINES : (# 3AIP37  # 1AIP37)
MEMBER : IP37
FIG-TYPE :
   CLOSED :  (# 3AIP37  # 2AIP37  #1AIP37)
ELEMENTS :   (# 1AIP37  # 2AIP37  # 3AIP37)
SPATIAL-RELATIONS :
   ABOVE-OF : ((# 1AIP37  # 2AIP37) (# 1AIP37 # 3AIP37)
                                    (# 2AIP37 # 3AIP37))
   LEFT-OF  : ((# 1AIPP37  # 2AIP37) ( # 3AIP37  #2AIP37))
ISA : SET
```

Figure 4.2

```
Structure  IP37

# -OF-SIDES-LEFT-OF-COMPOSED-OF :

      3-LEFT-OF-CURVED-LINES :
                          AIP37 : ((#1AIP37  # 2AIP37))
                          DIP37 : ((#2DIP37  # 3DIP37))
                          EIP37 : ((#1EIP37  # 3EIP37))

      4-LEFT-OF-CURVED-LINES :
                          AIP37 : ((# 3AIP37  # 2AIP37))
                          FIP37 : ((# 2FIP37  # 3FIP37))

COMPOSED-OF-LEFT-OF-# -OF-SIDES :

    CURVED-LINES-LEFT-OF-B :
                          BIP37 : ((# 2BIP37  # 18IP37))
                          CIP37 : ((# 3CIP37  # 2CIP37))

    CURVED-LINES-LEFT-OF-4 :
                          BIP37 : ((#2BIP37   # 3BIP37))

# -OF-SIDES-ABOVE-OF-COMPOSED-OF :

     3-ABOVE-OF-CURVED-LINES :
                          AIP37 : ((#1AIP37   # 2AIP37))
                          BIP37 : ((# 1BIP37  # 2BIP37))
                          CIP37 : ((#2CIP37   # 3CIP37))
                          DIP37 : ((# 2DIP37  # 3DIP37))
                          EIP37 : ((#1EIP37   # 3EIP37))

     4-ABOVE-OF-CURVED-LINES :
                          CIP37 : ((#1CIP37   # 3CIP37))
                          DIP37 : ((#1DIP37   # 3DIP37))
                          EIP37 : ((#2EIP37   # 3EIP37))
                          FIP37 : ((#2FIP37   # 3FIP37))

COMPOSED-OF-ABOVE-OF-# -OF-SIDES :

    CURVED-LINES-ABOVE-OF-3 : NIL

    CURVED-LINES-ABOVE-OF-4 :
                          AIP37 : ((#2AIP37   # 3AIP37))
                          BIP37 : ((# 2BIP37  # 3BIP37))
```

knowledge is box-independent and class conjectures can be readily made by inspecting the contents of such slots.

The reason behind representing spatial information in this way lies in the answer to the following question: 'When and how should spatial relations be inspected for patterns involving their examples during the problem solving process?' In the current implementation of the program, this is solved in the following manner. So far we have considered two methods of combining features, namely by specialisation and generalisation. So whenever a task is selected which suggests combining a feature with some other feature the two former methods were tried out. Now the program will consider a new possibility of relating two features which is whether the two features are related by any spatial relation. In other words, investigate whether any spatial relation, left-of for example, connects two figures such that the first one is described by feature f(1) and the second by feature f(2), where f(1) and f(2) are the two features being combined.

The next chapter provides the complete solution to problem 37 where the rule involves a spatial relation.

Let the slot f(i)-r(k)-f(j) where f(i), f(j) are features and r(k) is a spatial relation be defined as a feature-relation. Its aspects namely (a(i)-r(k)-a(j) |

for all i,j a(i), a(j) ε A(i) ∪ A(j)) are termed as aspect-relations. For example, number-of-sides-above-of-sketch is a feature-relation and 3-above-of-outline is an aspect-relation associated with it.

A heuristic useful in investigating feature-relations parallel to the second heuristic in the first section of this chapter is the following:

> For each aspect-relation a(i) of the feature-relation f
> IF a(i) occurs in every box belonging to class c
> THEN add the following task to the agenda:
> Conjecture: Every box in class c has aspect-relation a(i) of feature-relation f.

This heuristic is useful in problems 36, 37, 47 and 48 among others (see appendix 2). We conclude this brief analysis of rules involving spatial relations and turn to the computation of feature and feature-relation ratings.

4.4 Evaluation functions

In this section we provide answers to the following questions:

(i) Why are worths necessary?

(ii) How are the worths of features and feature-relations actually computed?

So far in our discussion the need for worths for features and feature-relations has not been motivated. We illustrate here that the crux of the program's problem-solving ability lies in the capacity to select features and feature-relations from a list of possibilities using their worth as an index of their importance.

The function used to compute the worth of a feature f is given below:

$$\text{worth}(f) = (1/|A(f)|) \times \sum_{\substack{\text{over all} \\ \text{boxes in } c}} (\text{number of figures in box } b \text{ possessing aspect a of feature } f)/(\text{total number of figures in } b)$$

where $A(f)$ is the set of aspects of f which actually occur in the class.

An example will illustrate the computation of worths by the above formula. Consider problem 28 (see figure 3.1). We assume that the worths of the features are to be calculated over the boxes of class I.

(i) Feature - area

$|A(\text{area})| = 1$ as all figures in class I are small.

For all the boxes in class I,

(number of figures possessing small area)/(total number of figures in box) = 1

Hence the worth (area) = 10 x 6 = 60.

(ii) Feature - shape.

The calculation of worth (shape) is the same as (i) since all the figures in class I are convex.

Hence worth (shape) = 60.

(iii) Feature - figure-type

Worth (figure-type) = 60 as all the figures in class I are closed.

(iv) Feature - composed-of

|A(composed-of)| = 2 as class I contains straight-line figures and

curvilinear figures. Also since in every box in I,

Number of straight-line figures + number of curvilinear figures is = total number of figures in the box,

we have worth (composed-of) = 1/2(10 x 6) = 30.

(v) Feature - sketch.

The calculation is the same as in (iv), so the

worth (sketch) = 30.

(vi) Feature - number-of-sides.

|A(number-of-sides)| = 1 as class I contains only three-sided figures. In box A, number of three-sided figures = 1, and the total number of figures in A is 4 so the first term in the summation is (1/4) x 10 = 2.5

Repeating this calculation for all the boxes in class I we obtain worth (number-of-sides) =

(10/4 + 20/6 + 0 + 30/6 + 30/6 + 0) = about 16.

Three types of worth calculations have been demonstrated. They are: (i) Universal type: Features like area in problem 28, which are constant over all the boxes and figures. These features are given the highest worth by the above function.

(ii) Binary type: Features like sketch in problem 28 have two aspects and all the figures in each box in the class can be described by one of these two aspects. These are considered moderately interesting.

(iii) Almost random type: Features like number-of-sides in problem 28 are examples of this type. Not all the figures in a box can be described by them and they do not occur in all the boxes. For example: In problem 28, in boxes C and E in class I, no three-sided figures occur. Accordingly these features are rated the lowest.

Based on the preceding observations, we can give a justification for the behaviour of the feature evaluation function.

(i) Rates Universal features highly: An investigation of the aspect examples of this type either produce rule yielding conjectures or result in the elimination of the feature. Both situations are very useful from the point of view of solving the problem.For example:Consider the trace of problem 28

shown in the last chapter. The features area, figure-type and shape get the highest worth and are accordingly investigated first by the individual feature heuristics. All three produce conjectures which are invalid rule candidates, but concomitantly all of them are eliminated from further consideration resulting in a drastic pruning of the search space.

(ii) Rates 'almost random' features lowly: Again, the example of problem 28 illustrates the correctness of this strategy. The feature number-of-sides is given a low priority and is hence never considered at all. As the solution to the problem demonstrates, number-of-sides was indeed irrelevant to the rule.

(iii) Rates binary types between (i) and (ii) : This enables the individual feature heuristics to first eliminate irrelevant features if any, and concentrate on those remaining features which are not also random types.

This feature evaluation strategy seems to be empirically correct as it was very successful on numerous problems solved using the program. Before concluding the discussion on feature evaluation functions, we describe the process of feature elimination in more detail. The simplest strategy to eliminate features from consideration by other heuristics is to reduce their worth to zero. Clearly this must be

done only in those cases in which the irrelevance of the feature to the rule is firmly established, as for example in the case of universal feature types generating conjectures which are invalid rule candidates.

A more sophisticated strategy would increase or decrease the worth of the feature dynamically, during the problem solving process, depending on whether the feature is found to be more or less relevant to the rule, than it was originally thought. A version of this was originally implemented in the program, but was abandoned later since the results of such a strategy seem difficult to predict. In fact, in some cases the effect was almost disastrous as the worth of some of the features started varying monotonically.

We now turn to the computation of worths of feature-relations. The evaluation function which assigns worths to feature-relations is given below. Given a feature-relation f, its worth is computed as

worth (f) $= (1/|A(f)|) \times \sum_{\text{all boxes in c}} |\text{figure-tuples (f)}| \times 10/C(\text{card(b)}, 2)$ where $|A(f)|$ = number of aspect-relations of f in class c.

and $|\text{figure-tuples (f)}|$ = number of pairs (f(1),f(2)) of figures such that f(1) is related to f(2) by an aspect-relation of f in box b, card (b) = number of figures in b.

Also C(n,r) = number of combinations of n things taken r at a time.

Intuitively, each summation term gives the fractional number of figure pairs related by the feature-relation f, of the total possible number of such pairs in a box.

An example will illustrate the computation of feature-relation worths by the above function.

Consider the feature-relation # -of-sides-left-of-composed-of depicted in structure I, problem 37 in figure 4.2. We have |A(# -of-sides-left-of-composed-of)| = 2 since in class I there are three-sided figures left of curvilinear figures and four-sided figures left of curvilinear ones. Also for box A, |figure-tuples (f)| = 2, as there is one three sided figure left of a curvilinear figure and one four-sided figure left of a curvilinear one in box A. Hence since |card (A)| = 2, the first term of the summation is = 10 x 2/3 = 6.67. Computing the remaining terms in the same way we obtain, worth(#-of-sides-left-of-composed-of) = (10/2) x (2/3 + 1/3 + 1/3 + 1/3) = 8.33

Since the feature-relation does not occur in all the boxes in class I and in every box in which it does, except in A, it does not obtain in more than one pair, thus it has an almost random character and its low worth is justified.

In summary, the following remarks may be made in connection with the above evaluation functions.

(i) The exact form of these functions is not important. Any function which behaves in the same way towards the three feature types will do the job.

(ii) Since the worth of a task depends on the worth of its arguments, which are frequently features or feature-relations, the problem-solving ability of the program depends to a large part on these functions.

4.5 The agenda mechanism

In this section, we describe a few details of the agenda control mechanism which were briefly mentioned in the last chapter.

Task ratings

As has been described earlier, when a task is suggested, it is added to the agenda after computing its worth. Here we describe this process of worth computation of tasks in more detail.

Usually, when a heuristic suggests a task, it also assigns a worth to it based on some special considerations. For example:The task 'Fill-in examples of the intersection of the aspects a(1) and a(2) of features f(1) and f(2) in class c' is assigned an initial worth = average (worth (f(1)), worth (f(2))). But this task is not added to the agenda with this initial worth for the following reason.

''The final worth of a task must depend on the nature of the task.''

To justify this assertion consider the following argument:

1. Tasks representing conjectures are the most important and should be executed as soon as possible since there is a possibility that they may turn out to be valid rule candidates or result in a pruning of the search space. In the worst case, they add tasks to the agenda thus widening the breadth of the search.

2. Tasks involving checking examples of features for patterns should be considered next in the scale of importance. If successful, they add conjectures to the agenda. In the worst case nothing happens.

3. Tasks which add new slots to class structures can be treated last as these always result in a increase in the size of the search space.

The above considerations are reflected in the assignations of final worths of tasks when they are added to the agenda. That is, Final-worth (task) = Initial-worth (task) (1 + k) where k is a function from a task-type to a non-zero number. For the types of tasks illustrated earlier, k values are given below.

TASK-TYPE	k VALUE
1. Conjecture	0.5
2. Check	0.3
3. Fill-in	0.2

In general, these may be any number of task-types. Thus for the above cases, if a conjecture, check and fill-in are proposed with the initial worth 30, the conjecture gets a final worth of 45, the check 39 and the fill-in 36. Thus conjectures are treated partially by the program.

Possibility lists

Consider the following general problem :

A certain computation has to be performed on a list of competing data items a(1), a(2),... a(n). For an example of this consider the following task: Specialise the aspect curvilinear of the feature composed-of in class I. Here the computation to be repetitively performed is that of selecting a feature other than that of composed-of and running heuristics on the pair (composed-of, feature (k)) to determine whether they should be combined. The problem is to determine a method of incorporating this iterative process into the agenda mechanism.

Strategy 1: Implement it as n copies of the task - 'Do X on a(i)' This simple scheme has the disadvantage that it usually leads to an explosive situation with n(1)

copies of task 1,n(2) copies of task 2 and so on, giving the agenda mechanism a breadth-first search character.

Strategy 2: A more complex strategy is the following one used by Lenat's AM program [LENAT 76] . Implement it as just one burst of computation which repeats the computation on the data items a(1), a(2),... . The amount of time that should be spent totally for this task is allotted initially. So the computation performed by the task is measured by a special parameter clause. When the amount of space/time allotted is over the computation ceases. The success of this scheme depends on the ability to allot space/time requirements apriori.

Stragegy 3 : The scheme implemented in our program is midway in terms of complexity between 1 and 2.

We introduce a new type of task called a possibilities list, which typically has the following structure : 'Do x on a(i). Remaining elements are a(i + 1),...a(n). Current task worth = f(a(i)) 'with the following interpretation : When the task is initially selected, do x on the first element. Then select the next task from the agenda but reattach the previous task after decrementing the list of data items by the one just completed.

The success of this scheme depends on whether the worth of the task can be expressed as a function of the

current data element. As is shown in the next chapter, this scheme is implemented for four types of iterative tasks, namely checking single features, specialising and generalising aspects and determining feature-relations.

We conclude this chapter on problem-solving methodology and in the next demonstrate traces of the program in execution on two sample problems. Most of the points discussed here will be illustrated by specific example there.

CHAPTER 5

ILLUSTRATING THE METHODOLOGY

In this chapter we present two more examples of solutions to Bongard problems as generated by the program. The problems being solved are 13 and 37 (for an illustration of these problems, see appendix 2).

Problem 37 is an instance where the rule involves the spatial distribution of figures in a box. The trace should exemplify the methodology used to discover rules involving spatial relations between figures in a box.

In problem 13, we have illustrated an instance where both specialization and generalization of features are important to rule discovery.

5.1 SOLUTION TRACE OF PROBLEM 13

CYCLE # 0

1 TASK(S) ON AGENDA

Initialising the list (FIGURE-TYPE COMPOSED-OF SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) of individual features for investigation in class IP13

CYCLE # 1

1 TASK(S) ON AGENDA

Investigating the feature FIGURE-TYPE for patterns in its examples in class IP13

CYCLE # 2

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP13 is CLOSED

...testing the conjecture on the boxes of class IIP13

conjecture is an invalid rule candidate for class IP13

since it is true of the boxes (FIIP13 EIIP13 DIP13 CIP13 BIIP13 AIIP13) in class IIP13

this feature is being eliminated from the list of features relevant to the rule

---

CYCLE # 3

1 TASK(S) ON AGENDA

Investigating the feature SHAPE for patterns in its examples in class IP13

CYCLE # 4

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP13 is CONVEX

...testing the conjecture on the boxes of class IIP13

conjecture is an invalid rule candidate for class IP13

since it is true of the boxes (FIIP 13 EIIP13 DIIP13 CIIP13 BIIP13 AIIP13) in class IIP13

this feature is being eliminated from the list of features relevant to the rule

---

CYCLE # 5

1 TASK(S) ON AGENDA

Investigating the feature SKETCH for patterns in its examples in class IP13

CYCLE # 6

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP13 is OUTLINE

...testing the conjecture on the boxes of class IIP13

conjecture is an invalid rule candidate for class IP13

since it is true of the boxes (FIIP13 EIIP13 DIIP13 CIIP13 BIIP13 AIIP13) in class IIP13

this feature is being eliminated from the list of features relevant to the rule

---

CYCLE # 7

1 TASK(S) ON AGENDA

Investigating the feature COMPOSED-OF for patterns in its examples in class IP13

CYCLE # 8

4 TASK(S) ON AGENDA

Conjecture : Feature COMPOSED-OF is single valued over the boxes of class IP13

testing the conjecture on the boxes of class IIP13

...conjecture is an invalid rule candidate for class IP13 since it is true of the boxes (AIIP13 BIIP13 CIIP13 DIIP13 EIIP13 FIIP13) in class IIP13

CYCLE # 9

3 TASK(S) ON AGENDA

Trying to form combinations of features involving aspect CURVILINEAR of feature COMPOSED-OF in class IP13

CYCLE # 10

6 TASK(S) ON AGENDA

Trying to form combinations of features involving aspect STRAIGHT-LINE of feature COMPOSED-OF in class IP13

CYCLE # 11

9 TASK(S) ON AGENDA

Filling in examples of the intersection of CURVILINEAR figures and HORIZONTAL figures in class IP13

CYCLE # 12

9 TASK(S) ON AGENDA

Checking the feature COMPOSED-OF-AND-AXIS for patterns involving its examples in class IP13

CYCLE # 13

9 TASK(S) ON AGENDA

Trying to form combinations of features involving aspect CURVILINEAR-AND-HORIZONTAL of feature COMPOSED-OF-AND-AXIS in class IP13

CYCLE # 14

13 TASK(S) ON AGENDA

Filling in examples of the union of CURVILINEAR figures and STRAIGHT-LINE figures in class IP13

CYCLE # 15

12 TASK(S) ON AGENDA

Filling in examples of the intersection of STRAIGHT-LINE figures and FOUR-SIDED figures in class IP13

CYCLE # 16

12 TASK(S) ON AGENDA

Filling in examples of the union of STRAIGHT-LINE figures and HORIZONTAL figures in class IP13

CYCLE # 17

11 TASK(S) ON AGENDA

Filling in examples of the union of CURVILINEAR-AND-HORIZONTAL figures and STRAIGHT-LINE figures in class IP13

CYCLE # 18

10 TASK(S) ON AGENDA

Investigating the feature NUMBER-OF-SIDES for patterns in its examples in class IP13

CYCLE # 19

11 TASK(S) ON AGENDA

Trying to form combinations of features involving aspect FOUR-SIDED of feature NUMBER-OF-SIDES in class IP13

CYCLE # 20

14 TASK(S) ON AGENDA

Filling in examples of the intersection of FOUR-SIDED figures and VERTICAL figures in class IP13

CYCLE # 21

13 TASK(S) ON AGENDA

Checking the feature NUMBER-OF-SIDES-AND-AXIS for patterns involving its examples in class IP13

CYCLE # 22

14 TASK(S) ON AGENDA

Trying to form combinations of features involving aspect FOUR-SIDED-AND-VERTICAL of feature NUMBER-OF-SIDES-AND-AXIS

CYCLE # 23

17 TASK(S) ON AGENDA

Filling in examples of the union of FOUR-SIDED figures and CURVILINEAR-AND-HORIZONTAL figures in class IP13

CYCLE # 24

16 TASK(S) ON AGENDA

Filling in examples of the intersection of FOUR-SIDED-AND-VERTICAL figures and STRAIGHT-LINE figures in class IP13

CYCLE # 25

16 TASK(S) ON AGENDA

Checking the feature NUMBER-OF-SIDES-AND-AXIS-AND-COMPOSED-OF for patterns involving its examples in class IP13

CYCLE # 26

16 TASK(S) ON AGENDA

Trying to form combinations of features involving aspect FOUR-SIDED-AND-VERTICAL-AND-STRAIGHT-LINE of feature NUMBER-OF-SIDES-AND-AXIS-AND-COMPOSED-OF in class IP13

CYCLE # 27

20 TASK(S) ON AGENDA

Filling in examples of the union of FOUR-SIDED-AND-VERTICAL figures and CURVILINEAR-AND-HORIZONTAL figures in class IP13

CYCLE # 28

20 TASK(S) ON AGENDA

Checking the feature NUMBER-OF-SIDES-AND-AXIS-OR-COMPOSED-OF-AND-AXIS for patterns involving its examples in class IP13

CYCLE # 29

20 TASK(S) ON AGENDA

Conjecture : Every figure in class IP13 is FOUR-SIDED-AND-VERTICAL-OR-CURVILINEAR-AND-HORIZONTAL

...testing the conjecture on the boxes of class IIP13

Conjecture is a valid rule candidate for class IP13

TASK : STATE-RULE (IP13 NUMBER-OF-SIDES-AND-AXIS-OR-COMPOSED-OF-AND-AXIS FOUR-SIDED-AND-VERTICAL-OR-CURVILINEAR-AND-HORIZONTAL)

RULE: CLASS IP13 FEATURE NUMBER-OF-SIDES-AND-AXIS-OR-COMPOSED-OF-AND-AXIS HAS-VALUE FOUR-SIDED-AND-VERTICAL-OR-CURVILINEAR-AND-HORIZONTAL

5.2 SOLUTION TRACE OF PROBLEM 37

CYCLE # 0

1 TASK(S) ON AGENDA

Initialising the list (FIGURE-TYPE COMPOSED-OF SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) of individual features for investigation in class IP37

CYCLE # 1

1 TASK(S) ON AGENDA

Investigating the feature FIGURE-TYPE for patterns in its examples in class IP37

CYCLE # 2

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP37 is CLOSED

...testing the conjecture on the boxes of class IIP37

conjecture is an invalid rule candidate for class IP37

since it is true of the boxes (FIIP37 EIIP37 DIIP37 CIIP37 BIIP37 AIIP37) in class IIP37

this feature is being eliminated from the list of features relevant to the rule

---

CYCLE # 3

1 TASK(S) ON AGENDA

Investigating the feature SHAPE for patterns in its examples in class IP37

CYCLE # 4

2 TASK(S) ON AGENDA

Conjecture : Every figure in class IP37 is CONVEX

...testing the conjecture on the boxes of class IIP37

conjecture is an invalid rule candidate for class IP37

since it is true of the boxes (FIIP37 EIIP37 DIIP37

CIIP37 BIIP 37 AIIP37) in class IIP37

this feature is being eliminated from the list of features

relevant to the rule

CYCLE # 5

1 TASK(S) ON AGENDA

Investigating the feature SKETCH for patterns in its

examples in class IP37

CYCLE # 6

2 TASK(S) ON AGENDA

Conjecture: Every figure in class IP37 is OUTLINE

...testing the conjecture on the boxes of class IIP37

conjecture is an invalid rule candidate for class IP37

since it is true of the boxes (FIIP37 EIIP37 DIIP37

CIIP37 BIIP37 AIIP37) in class IIP37

this feature is being eliminated from the list of features

relevant to the rule

CYCLE # 7

1 TASK(S) ON AGENDA

Investigating the feature AREA for patterns in its examples in class IP37

CYCLE # 8

2 TASK(S) ON AGENDA

Conjecture: Every figure in class IP37 is SMALL

...testing the conjecture on the boxes of class IIP37

conjecture is an invalid rule candidate for class IP37

since it is true of the boxes (FIIP37 EIIP37 DIIP37 CIIP37 BIIP37 AIIP37) in class IIP37

this feature is being eliminated from the list of features relevant to the rule

---

CYCLE # 9

1 TASK(S) ON AGENDA

Investigating the feature COMPOSED-OF for patterns in its examples in class IP37

CYCLE # 10

4 TASK(S) ON AGENDA

Conjecture: There is a STRAIGHT-LINE figure in every box in class IP37

Testing it on the boxes of class IIP37

...conjecture is an invalid rule candidate for class IP37

since it is true of boxes (FIIP37 EIIP37 DIIP37 CIIP37 BIIP37 AIIP37) in class IIP37

CYCLE # 11

5 TASK(S) ON AGENDA

Conjecture : There is a CURVILINEAR figure in every box in class IP37

Testing it on the boxes of class IIP37

...conjecture is an invalid rule candidate for class IP37 since it is true of boxes (FIIP37 EIIP37 DIIP37 CIIP37 BIIP37 AIIP37) in class IIP37

CYCLE # 12

6 TASK(S) ON AGENDA

Conjecture : In every box in class IP37 examples of STRAIGHT-LINE figures are GREATER THAN examples of CURVILINEAR figures

Testing it on the boxes in class IIP37

...conjecture is an invalid rule candidate for class IP37 since conjecture is true of the boxes (AIIP37 BIIP37 CIIP37 DIIP37 EIIP37 FIIP37) of class IIP37

CYCLE # 13

5 TASK(S) ON AGENDA

Specialise the aspect STRAIGHT-LINE of the feature COMPOSED-OF in class IP37

CYCLE # 14

5 TASK(S) ON AGENDA

Checking spatial-relations for patterns involving their examples between STRAIGHT-LINE figures and others in class IP37

CYCLE # 15

9 TASK(S) ON AGENDA

Specialise the aspect CURVILINEAR of the feature COMPOSED-OF in class IP37

CYCLE # 16

9 TASK(S) ON AGENDA

Checking spatial-relations for patterns involving their examples between CURVILINEAR figures and others in class IP37

CYCLE # 17

9 TASK(S) ON AGENDA

Filling in examples of STRAIGHT-LINE figures LEFT-OF CURVILINEAR figures in class IP37

CYCLE # 18

9 TASK(S) ON AGENDA

Filling in examples of STRAIGHT-LINE figures LEFT-OF STRAIGHT-LINE figures in class IP37

CYCLE # 19

8 TASK(S) ON AGENDA

Filling in examples of STRAIGHT-LINE figures ABOVE-OF CURVILINEAR figures in class IP37

CYCLE # 20

8 TASK(S) ON AGENDA

Filling in examples of STRAIGHT-LINE figures ABOVE-OF STRAIGHT-LINE figures in class IP37

CYCLE # 21

7 TASK(S) ON AGENDA

Checking the feature-relation COMPOSED-OF-ABOVE-OF-COMPOSED-OF for patterns in its examples in class IP37

CYCLE # 22

8 TASK(S) ON AGENDA

Conjecture : Every box in class IP37 has STRAIGHT-LINE figures ABOVE-OF STRAIGHT-LINE figures

Testing the conjecture on the boxes of class IIP37

...conjecture is an invalid rule candidate for class IP37 since it is true of boxes (AIIP37 BIIP37 CIIP37 DIIP37 EIIP37 FIIP37) in class IIP37

CYCLE # 23

7 TASK(S) ON AGENDA

Conjecture : Every box in class IP37 has STRAIGHT-LINE figures ABOVE-OF CURVILINEAR figures

Testing the conjecture on the boxes of class IP37

...conjecture is an invalid rule candidate for class IP37 since it is true of boxes (BIIP37) in class IIP37

CYCLE # 24

6 TASK(S) ON AGENDA

Trying to determine spatial-relations between STRAIGHT-LINE figures and those described by any 1 of the aspects of (FIGURE-TYPE SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) features in class IP37

CYCLE # 25

6 TASK(S) ON AGENDA

Trying to determine spatial-relations between CURVILINEAR figures and those described by any 1 of the aspects of (FIGURE-TYPE SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) features in class IP37

CYCLE # 26

10 TASK(S) ON AGENDA

Filling in examples of CURVILINEAR figures LEFT-OF FOUR-SIDED figures in class IP37

CYCLE # 27

11 TASK(S) ON AGENDA

Filling in examples of CURVILINEAR figures LEFT-OF THREE-SIDED figures in class IP37

CYCLE # 28

10 TASK(S) ON AGENDA

Filling in examples of CURVILINEAR figures ABOVE-OF FOUR-SIDED figures in class IP37

CYCLE # 29

11 TASK(S) ON AGENDA

Filling in examples of CURVILINEAR figures ABOVE-OF THREE-SIDED figures in class IP37

CYCLE # 30

10 TASK(S) ON AGENDA

Investigating the feature NUMBER-OF-SIDES for patterns in its examples in class IP37

CYCLE # 31

12 TASK(S) ON AGENDA

Conjecture : There is a THREE-SIDED figures in every box in class IP37

Testing it on the boxes of class IIP37

...conjecture is an invalid rule candidate for class IP37 since it is true of boxes (FIIP37 EIIP37 DIIP37 CIIP37 BIIP37 AIIP37) in class IIP37

CYCLE # 32

13 TASK(S) ON AGENDA

Conjecture : There is a FOUR-SIDED figure in every box in class IP37

Testing it on the boxes of class IIP37

...conjecture is an invalid rule candidate for class IP37 since it is true of boxes (FIIP37 EIIP37 DIIP37 CIIP 37 BIIP37 AIIP37) in class IIP37

CYCLE # 33

14 TASK(S) ON AGENDA

Conjecture : In every box in class IP37 examples of THREE-SIDED figures are EQUAL examples of FOUR-SIDED figures

Testing it on the boxes in class IIP37

...conjecture is an invalid rule candidate for class IP37 since conjecture is true of the boxes (AIIP37 BIIP37 CIIP37 DIIP37 EIIP37 FIIP37) of class IIP37

CYCLE # 34

13 TASK(S) ON AGENDA

Specialise the aspect THREE-SIDED of the feature NUMBER-OF-SIDES in class IP37

CYCLE # 35

13 TASK(S) ON AGENDA

Checking spatial-relations for patterns involving their examples between THREE-SIDED figures and others in class IP37

CYCLE # 36

13 TASK(S) ON AGENDA

Specialise the aspect FOUR-SIDED of the feature NUMBER-OF-SIDES in class IP37

CYCLE # 37

13 TASK(S) ON AGENDA

Checking spatial-relations for patterns involving their examples between FOUR-SIDED figures and others in class IP37

CYCLE # 38

13 TASK(S) ON AGENDA

Trying to determine spatial-relations between THREE-SIDED figures and those described by any 1 of the aspects of (FIGURE-TYPE SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) features in class IP37

CYCLE # 39

15 TASK(S) ON AGENDA

Filling in examples of THREE-SIDED figures LEFT-OF FOUR-SIDED figures in class IP37

CYCLE # 40

15 TASK(S) ON AGENDA

Filling in examples of THREE-SIDED figures ABOVE-OF FOUR-SIDED figures in class IP37

CYCLE # 41

15 TASK(S) ON AGENDA

Trying to determine spatial-relations between FOUR-SIDED figures and those described by any 1 of the aspects of (FIGURE-TYPE SHAPE SKETCH AREA NUMBER-OF-SIDES AXIS) features in class IP37

CYCLE # 42

15 TASK(S) ON AGENDA

Checking the feature-relation NUMBER-OF-SIDES-ABOVE-OF-COMPOSED-OF for patterns in its examples in class IP37

CYCLE # 43

15 TASK(S) ON AGENDA

Conjecture : Every box in class IP37 has THREE-SIDED figures ABOVE-OF-CURVILINEAR figures

Testing the conjecture on the boxes of class IIP37

Conjecture is a valid rule for class IP37

TASK: STATE-RULE (IP37 NUMBER-OF-SIDES-ABOVE-OF-COMPOSED-OF THREE-SIDED-ABOVE-OF-CURVILINEAR)

RULE: CLASS IP37 FEATURE NUMBER-OF-SIDES-ABOVE-OF-COMPOSED-OF HAS-VALUE THREE-SIDED-ABOVE-OF-CURVILINEAR

CHAPTER 6

CONCLUSIONS

In this chapter we consider briefly the capabilities and limitations of our methodology for solving Bongard problems. We conclude the thesis with some suggestions for future work.

The main issues that are of interest are:

(i) The scope of the representation and heuristics.

(ii) The domain independent nature of the knowledge used in rule discovery.

(iii) Advantages and limitations of the program structure.

(iv) Potential for further work.

The issue of the scope of the representation and heuristics lies in the answer to the question. 'How many and what types of Bongard problems can be solved with the existing representation and set of heuristics?'

While designing the program, a set of thirty problems from the original hundred were used for trial runs. Our claim is that with appropriate and justifiable extensions to the representation and heuristics, practically all the problems in the original collection of one hundred problems lie within the solvable class.

As an indication of the validity of the above claim, we make the following observations:

(a) If a new Bongard problem is selected, the individual figures can be represented by choosing an appropriate set of descriptors from the set U of 2-D geometry descriptors.

(b) These descriptors, if not already present, can be added to the existing feature set as new slots in the figure, box and class structures.

(c) It follows from the present formulation of the problem, that a rule can be discovered only if the representation includes all the constituents, like features and relations, needed to construct that rule.

To elaborate on this point, consider the example of Bongard problem 50, illustrated in appendix 2. The rule is that the contents of every box in class I are symmetric with respect to the y axis. Since symmetry with respect to an axis is not a part of the existing set of descriptors used in our representation, to be able to solve this problem requires that the feature 'y-axis-symmetry' be included in the representation. We are not concerned here with the lower level processing needed to compute this feature from the representation of the 'raw' image, as for example a pixel representation. The parsimony or completeness of our choice of features is not the issue here (provided, of course,

they are reasonably chosen. To take an example, consider problem 13. While the choice of the features axis, number-of-sides and composed-of is reasonable, including features like 'feature-X' where feature-X describes those figures which are four-sided and have their axis vertical will not be a reasonable choice).

(d) It should be evident from our earlier account of our problem solving methodology that heuristics can be added to the system to enhance the capability of the program to discover rules involving new patterns. For example: To enable the system to recognize rules involving compound features, the appropriate heuristic need only be added. Similarly heuristics which investigated spatial relations enabled rules involving spatial relations to be discovered. Contrast this with traditional pattern recognition systems where such performance increments cannot be achieved so easily.

For some of the problems it might be that the representation and processing needed to discover the rule becomes very complex and unwieldy. However by selecting an appropriate set of descriptors for figures and relations between them, it should be possible to generate the rules using the methodology we have outlined. The limitations lie not along this direction but along a different one as explained later in this chapter.

Since the knowledge used to discover rules is largely independent of the particular domain, applications of this methodology to other domains for rule discovery is possible. One promising area seems to be the problem of cell classification. A preliminary analysis of one such problem (see appendix 4) shows that the cell classification problem can be solved using our methodology.

Features here would correspond to geometric properties of cells as well as various cell bodies found in them. A rule would correspond to a test needed to determine whether a particular cell is of type X or not, which could depend on the presence or absence of certain types of objects occuring in the cell. A cell here is analogous to a box in our case.

As we had mentioned earlier, one of the advantages of our methodology lies in the separation that exists between the descriptive part of the representation and the part consisting of the heuristics. Specifically since the declarative part of the knowledge representation can be expanded independently without affecting the procedural part and vice-versa (within reasonable limits), any additions needed in the problem solving capability of the program is easy to incorporate.

This organisation will be very important and useful when attempting to apply this methodology to other domains. The overall problem can be broken up into a simple core component and incremental layers of performance improvement about the core. So after the basic problem is solved (for example, in our case the discovery of rules involving individual features), extensions to the existing set-up is easily made (in our case rule discovery involving feature compositions and relations).

We now turn to the limitations of our methodology. The most important limitation is that with the existing set-up automatic modification or expansion of the knowledge base cannot be achieved. That is, methods by which some form of learning takes place is not possible at present. This would imply that the system should be able to synthesise new heuristics and thus improve its problem solving capability on its own.

The following changes need to be made:

(i) Presently heuristics are implemented as opaque pieces of LISP code. An alternative method is to represent heuristics also in some declarative format. This will enable heuristics to be modified and thus new heuristics can be derived from existing ones.

This is currently being pursued by the EURISKO research group [LENAT 83].

(ii) Too much knowledge has been encoded into a single heuristic. One modification that could be made is to split the existing set of heuristics into a large number of simple ones.

(iii) Presently tasks are proposed and rated by individual heuristics with no attempt at supplying reasons or justifications for the act. One change would involve collecting and categorizing reasons for task proposals and rating each reason independently. Then the worth of the task would be the terms of the worth of its supporting reasons [LENAT 76].

(iv) The structures we use to represent figures, boxes and classes are just simple versions of the frames knowledge representation scheme [MINSKY 75]. We have not used the word 'frame' since two important components of the frame theory, namely default slot values and procedural attachments to frames are not used by us. While it is not clear how the first technique could be used for problem solving, linking procedures to slots in the structures used in representing classes could be a useful addition to the existing implementation.

Future work lies along all the above directions. Further investigation of the Bongard problems definitely needs to be done as any such endeavour could yield results valuable to the overall theory of problem solving.

REFERENCES

1. BONGARD, M.M., Pattern Recognition (Spartan Books, Washington, D.C. 1970).

2. CHARNIAK, E., RIESBECK, C.K., and McDERMOTT, D.V., Artificial Intelligence Programming (Lawrence Erlbaum Associates, New Jersey) 1980.

3. DAVIS, R. and LENAT, D., Knowledge based systems in Artificial Intelligence (McGraw-Hill, New York 1982).

4. HOFSTADTER, D.R., Godel, Escher, Bach : An eternal golden braid. (Basic books Inc. 1979).

5. LENAT, D.B., The Nature of Heuristics, AI 19(2), 1980, pp 189-249.

6. LENAT, D.B., Theory formation by Heuristic search, AI 21, 1983 pp 31-59.

7. LENAT, D.B., EURISKO : A Program That Learns New Heuristics and Domain Concepts AI 21, 1983, pp 61-98.

8. LENAT, D.B., AM : an artificial intelligence approach to discovery in mathematics as heuristic search, Ph.D. thesis, CSD, Standford Univ., CS-STAN-76-570, AIM-286, 1976.

9. MICHALSKI, R.S., Theory and Methodology of Inductive learning, AI (20) pp 111-161.

10. MINSKY, M., A Framework for representing knowledge in 'The Psychology of Computer Vision', (McGraw-Hill, New York) 1975.

11. NILSSON, N.J., Principles of Artificial Intelligence (Tioga, Palo Alto, CA, 1980).

12. NILSSON, N.J., Learning Machines (McGraw-Hill, New York 1965).

13. NILSSON, N.J., Problem Solving methods in Artificial Intelligence (McGraw-Hill, New York 1971).

14. NARASIMHAN, R., On the description, generation, and description of classes of pictures in 'Automatic Interpretation and Classification of Images' (Academic Press, New York) 1969.

APPENDIX 1

LIST OF HEURISTICS

We list here the entire set of heuristics that were used for solving Bongard problems. The list is not complete in the sense new types of problems may require additional heuristics. We have categorised these into those connected with

(i) Individual features
(ii) Specialisation of features
(iii) Generalisation of features and
(iv) Spatial relations.

(i) Individual features:

1. For a feature f and class c,

if f has a single aspect in c, and every figure in c is described by f, then conjecture that every figure of c has aspect a of f.

2. For an aspect a of feature f and class c,

if every figure in c is not described by f, and a occurs in every box in c, then conjecture that every box of c contains a figure described by aspect a of f.

3. If a feature f has multiple aspects in c, then try to discover patterns in f by applying (2) on every aspect a of f.

4. If a feature f has multiple aspects in c, try to determine numeric relationships between examples of any two different aspects of f.

5. For a feature f and class c, if f has multiple aspects in c, but for every box in c, f has a single aspect, conjecture that f is single-valued over c.

6. For a feature f and class c, if f occurs only in some boxes in c, and it occurs in every figure for those boxes, then try to form doubletons of features involving f and investigate them.

7. For a feature f and class c, if f occurs only in some boxes in c and it occurs only in some figures in a box, reduce the worth of this feature since it has a random occurence.

8. If the worth of a feature f is modified, then recompute the worths of all tasks on the agenda involving f.

9. If a group G of features f satisfies (2) in class c, conjecture that every box in class c contains figures identical over G.

10. For aspects a_1 and a_2 of feature f, if the set of examples of a_1 is related to that of a_2 by the same numerical relation in both classes, reduce the worth of f slightly.

11. For an aspect a of f, if a occurs in every box in a class and in any box in the other class, try to specialise a.

12. For an aspect a of f if a occurs in every box in a class and in any box in the other class determine spatial relations between figures described by a and others in the class.

13. If a feature f occurs in every figure in both classes and f has a single aspect in both classes, reduce the worth of f to zero.

14. To check a conjecture about a class c involving aspect a of f, a quick method is to determine whether a occurs at all in the other class.

15. If f is a newly created feature whose aspect examples have just been filled in, it might be worthwhile to immediately investigate it for unexpected conjectures.

16. For a feature f and its aspects, if a task involving f exists on the agenda with w_1, and an identical task is reproposed with worth w_2, then alter the worth of the existing task to maximum (w_1, w_2).

Feature specialization

17. A conjunction of aspects a_1 and a_2 of feature f_1 and f_2 respectively, is interesting if (i) f_1 and f_2 have non-zero worths.

(ii) f_1 and f_2 have worths which differ by a factor less than 3.

(iii) intersection of examples of a_2 and a_3 is non-null.

(iv) all examples of a_1 are not examples of a_2.

(v) all examples of a_2 are not examples of a_1.

18. To specialise aspect a of feature f, choose the most interesting feature f' different from f, and for each aspect a' of f', apply (17) on the pairs (a,f),(a',f').

19. If two features f_1 and f_2 are being combined where f_1 and f_2 are themselves composite, if number-of-feature-parts (f_1) + number-of-feature-parts (f_2) > number of features having non-zero worth, then this composition is not interesting.

20. To specialise aspect a of feature f, form a possibilities list of other features which may form a useful combination with f, and apply (18) on this list.

21. If efforts at combining a of f with any aspect of f' failed, repropose this task with a different feature f'' such that
worth (f') $\geq$ worth (f'') $\geq$ worth (f^*) for all f^* other than f', f and f''.

Feature generalisation

22. A disjunction of aspects a_1 of f_1 and a_2 of f_2 is interesting if

(i) f_1 and f_2 have non-zero worths

(ii) example of $a_1 \not\Rightarrow$ examples of the union of a_1 and a_2.

(iii) worth (f_1) differs from worth (f_2) by less than 1.5.

(iv) all examples of a_1 are not examples of a_2.

23. To generalise aspect a of f, applying (20) on the pair (a,a'), where a' is an aspect of f itself, different from a, may yield useful conjectures.

24. To generalise aspect a of f form a possibilities list of other features which may form a useful disjunction with f and apply (20) on each aspect of the most interesting feature in the list.

25. If a generalisation of aspect a of feature f with feature f' fails, repropose this task using other features as generalisation candidates.

Relations

26. If the aspect-relation a of the feature-relation f occurs in every box in c, then conjecture that fact.

27. If, for every occurence of a relation R between two figures ; figure 1 and figure 2 in each box in a class, some feature f is common to all the figure 1's and some feature f' is common to all the figure 2's. Conjecture that in all boxes in c, figures described by f are spatially related by R to figures described by f'.

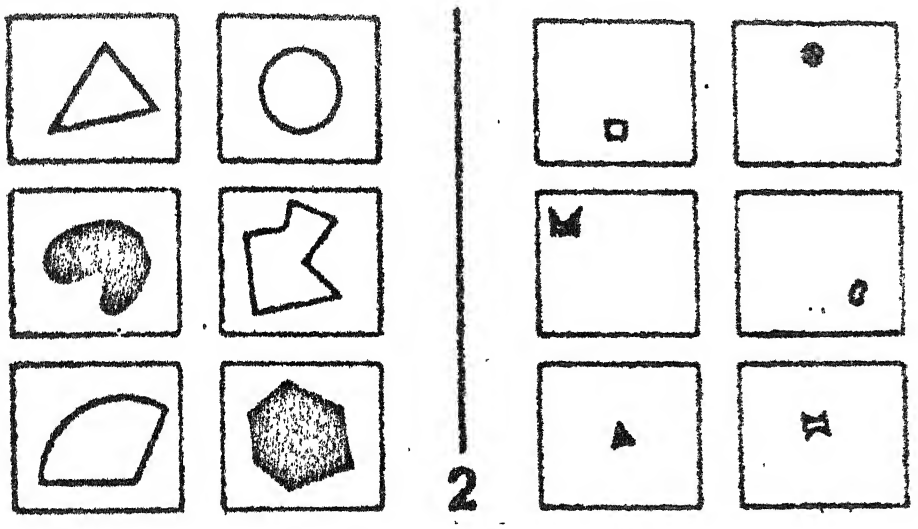
28. For a relation R and class **c**, if for every box in **c**, it is possible to find figure-pairs (figure 1, figure 2), (figure 3, figure 4), related by R such that figure 2 is identical to figure 3, then conjecture that R is a binary relation with a non-trivial transitive closure.

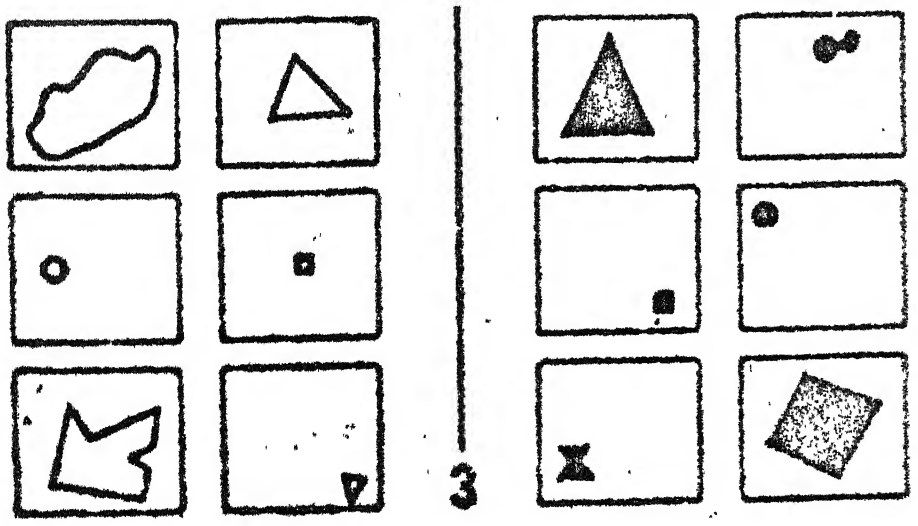
29. While filling in examples of the aspect-relation a-r-b of the feature-relation f-r-f', a useful strategy is to fill in b-r-a also, where r is a spatial relation and a and b are aspects of features f and f' respectively.

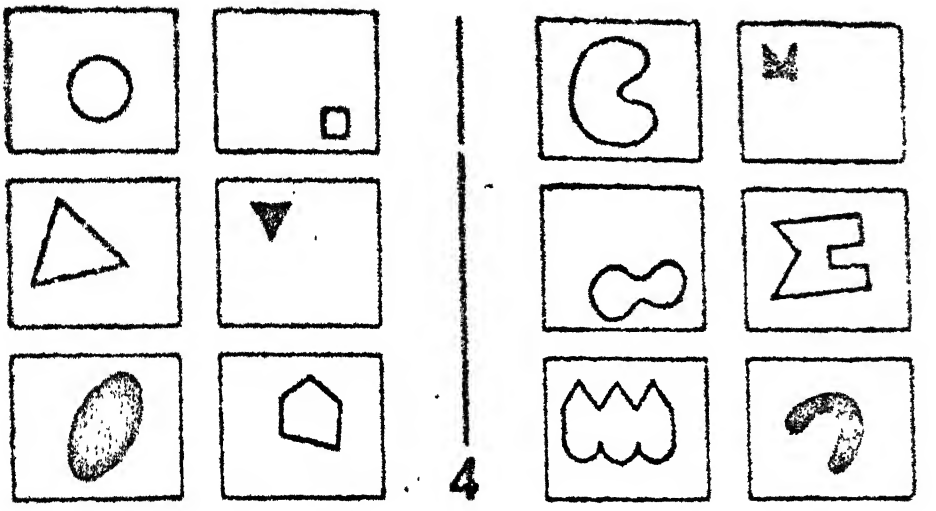
30. For a spatial relation R and class c for every box in c, try to determine numerical relationships between figures related by R and those which are not.

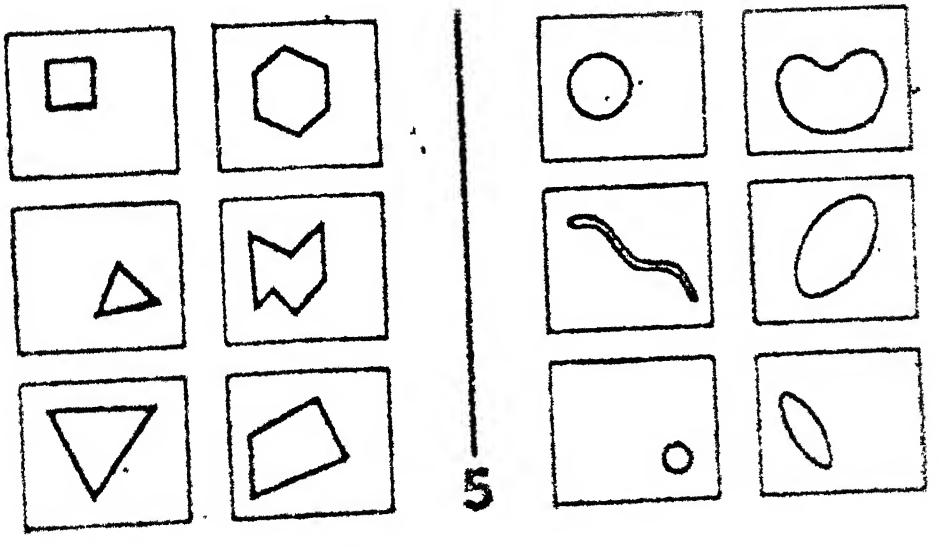
APPENDIX 2

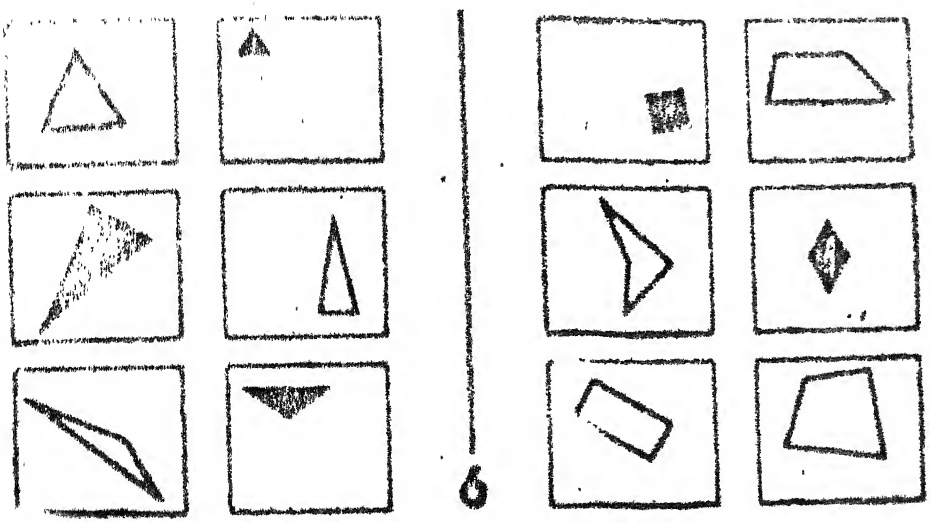
LIST OF SAMPLE BONGARD PROBLEMS

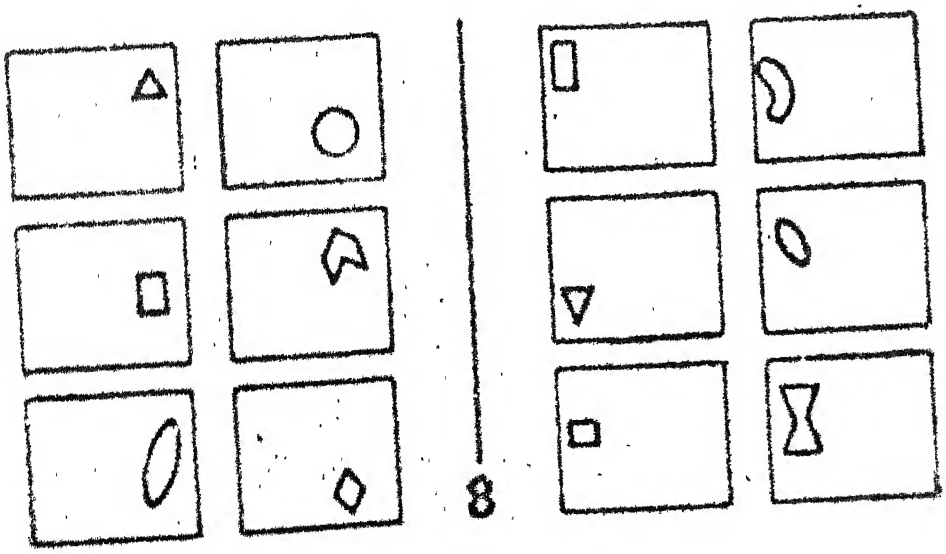


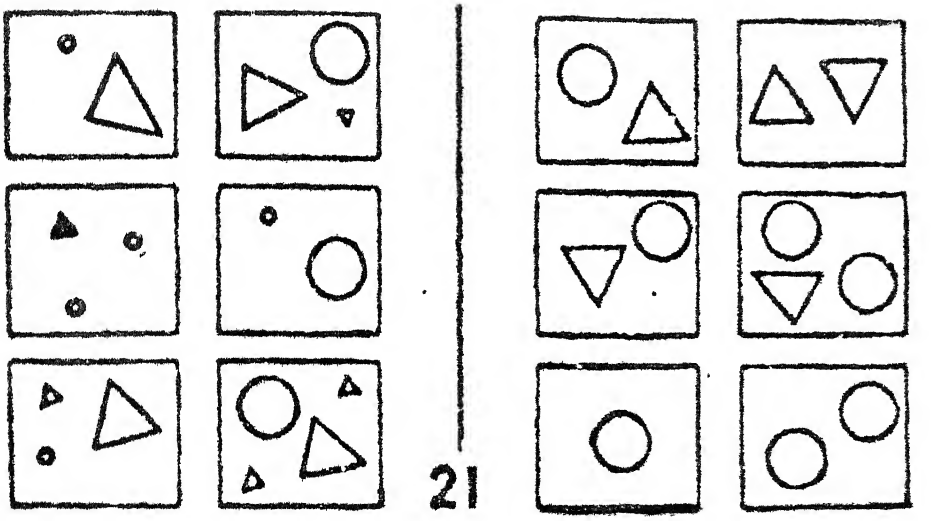


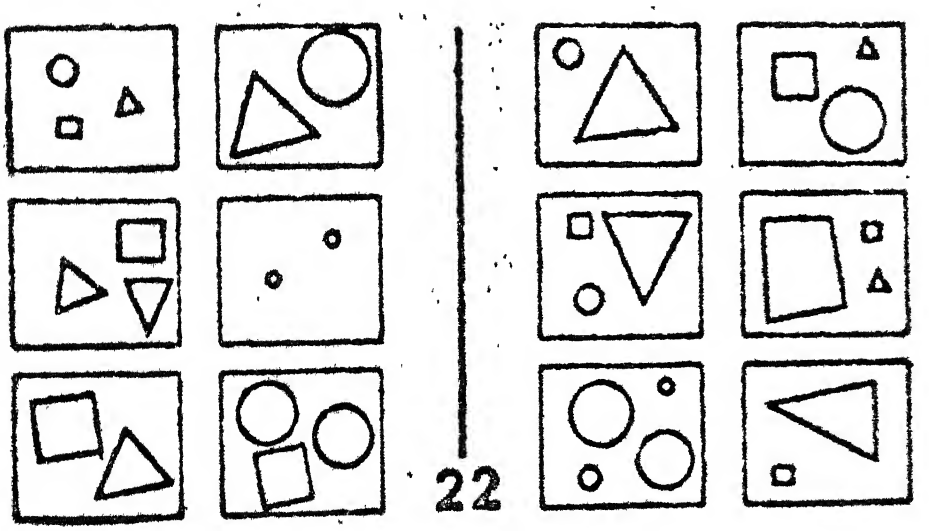

4

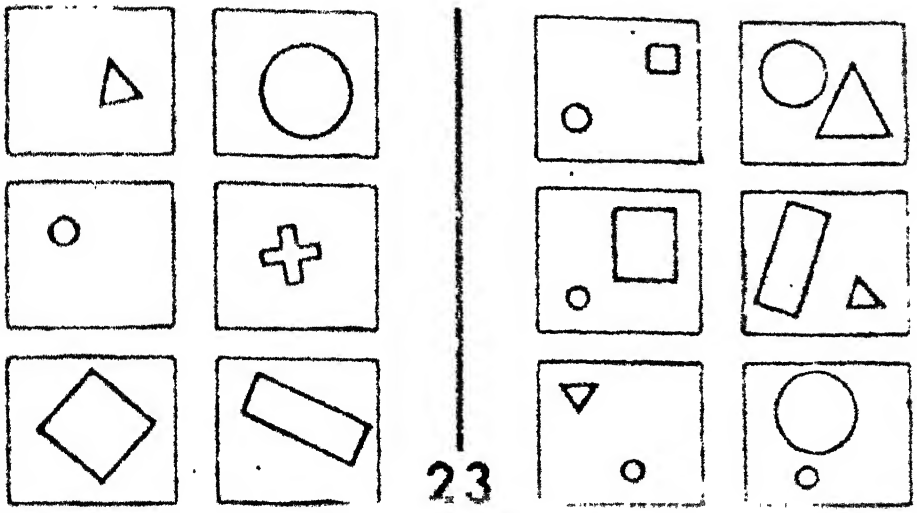

5

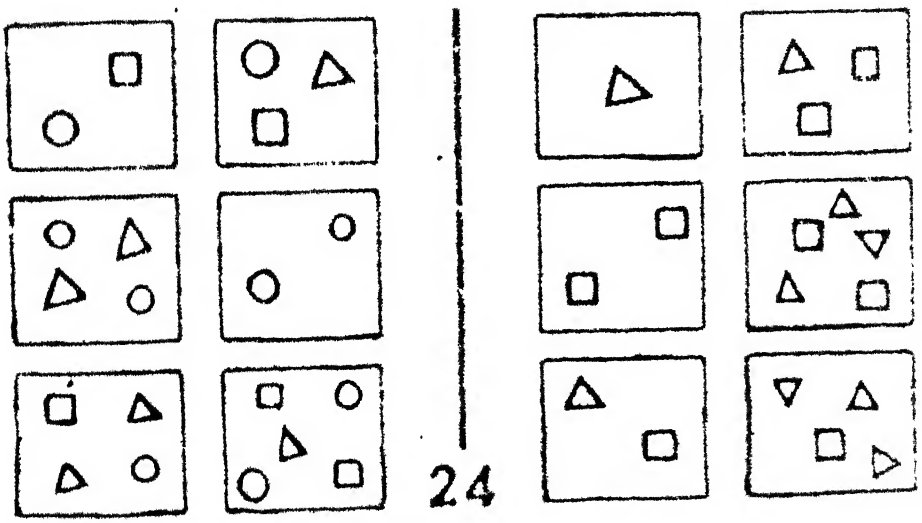

6

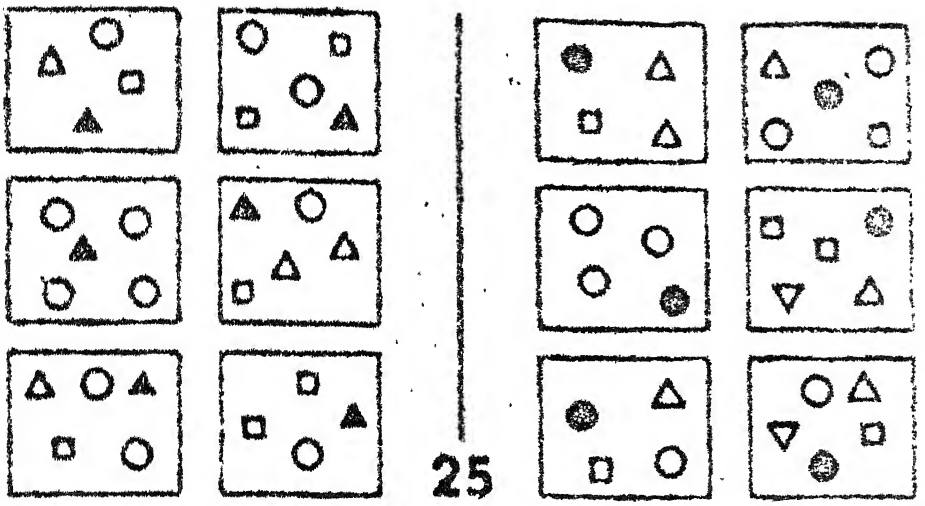

8


21

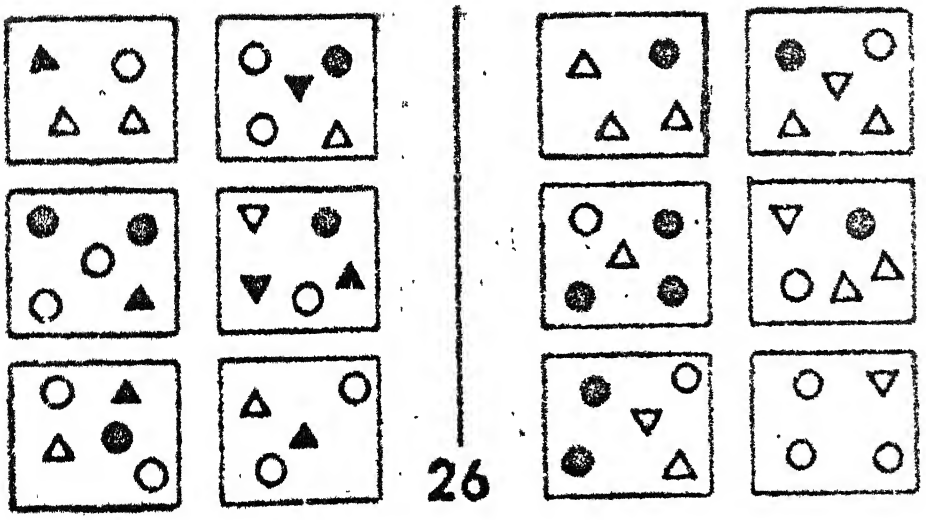

22


23

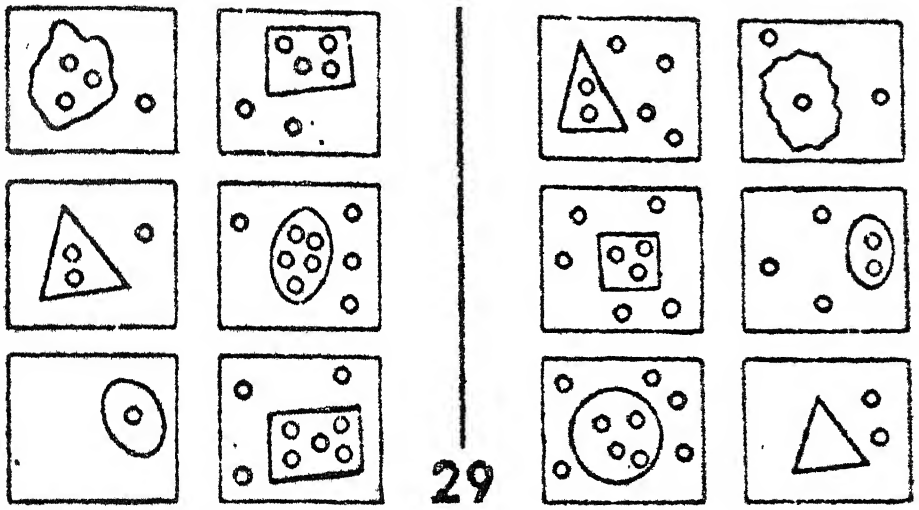

24



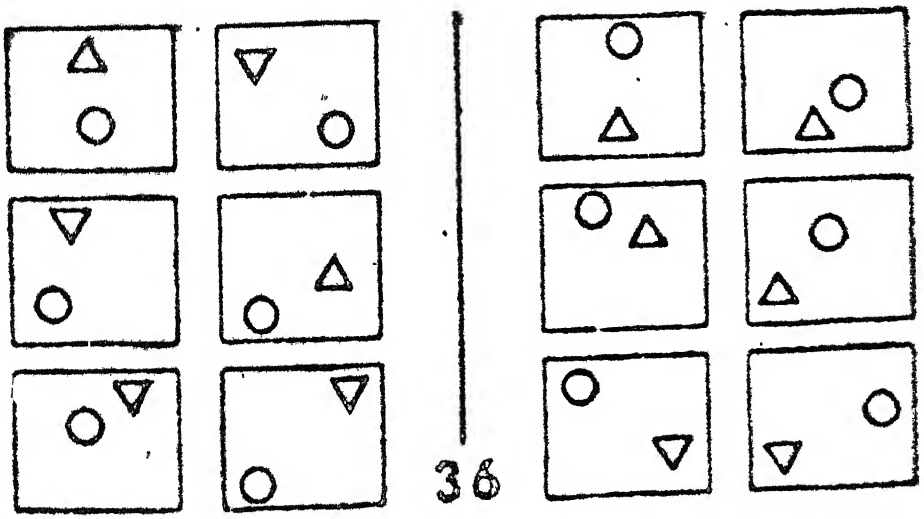
25



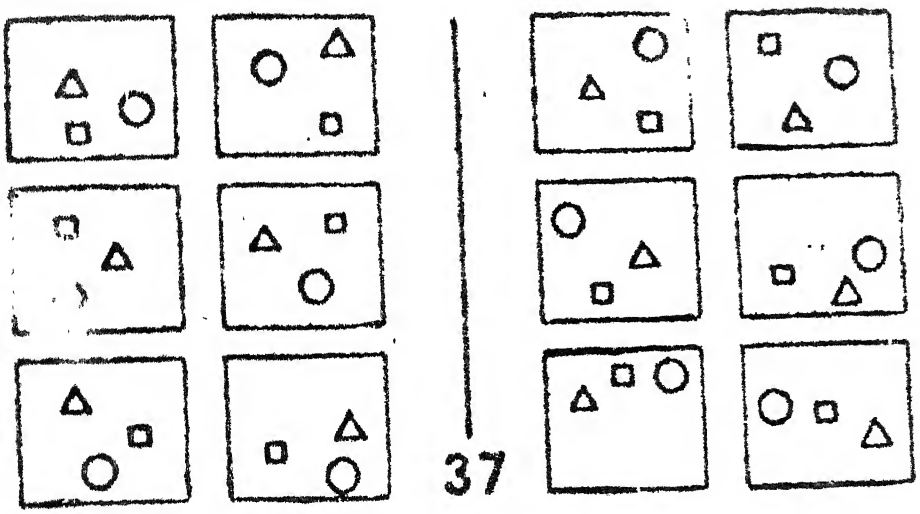
26

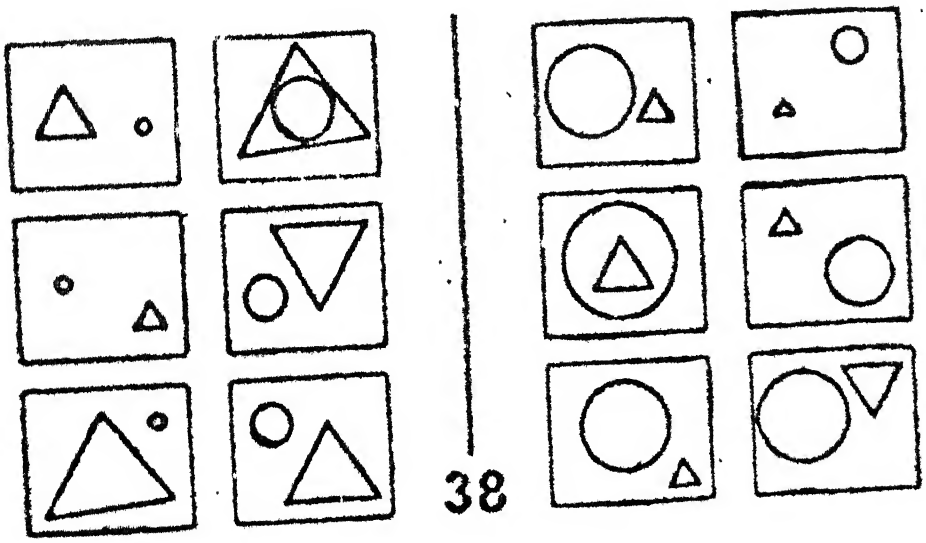


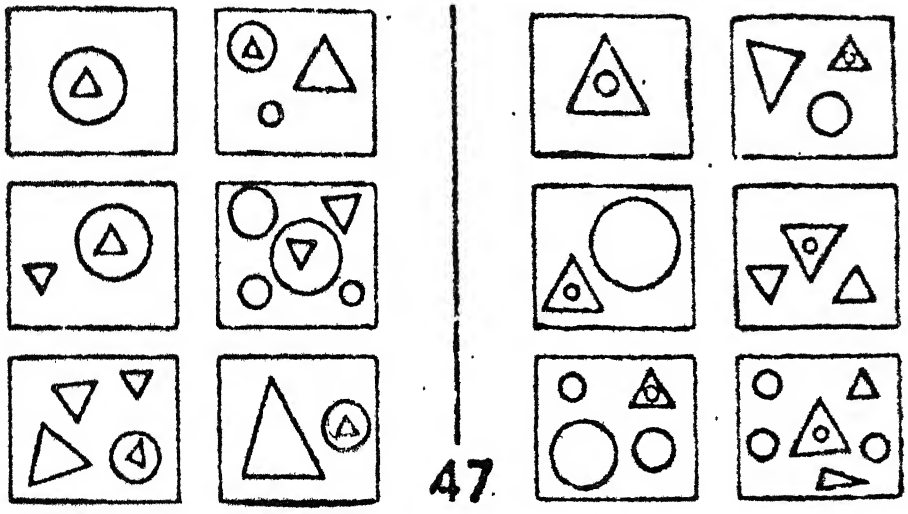
29



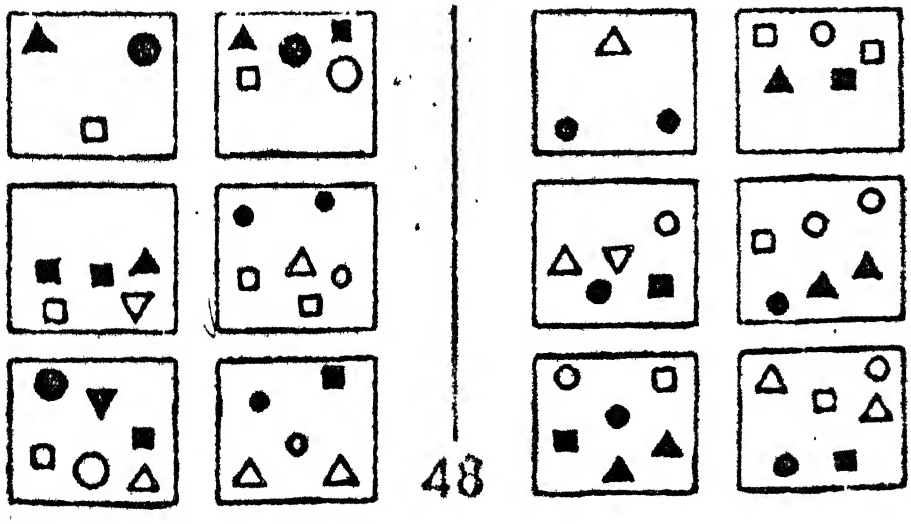
36


37

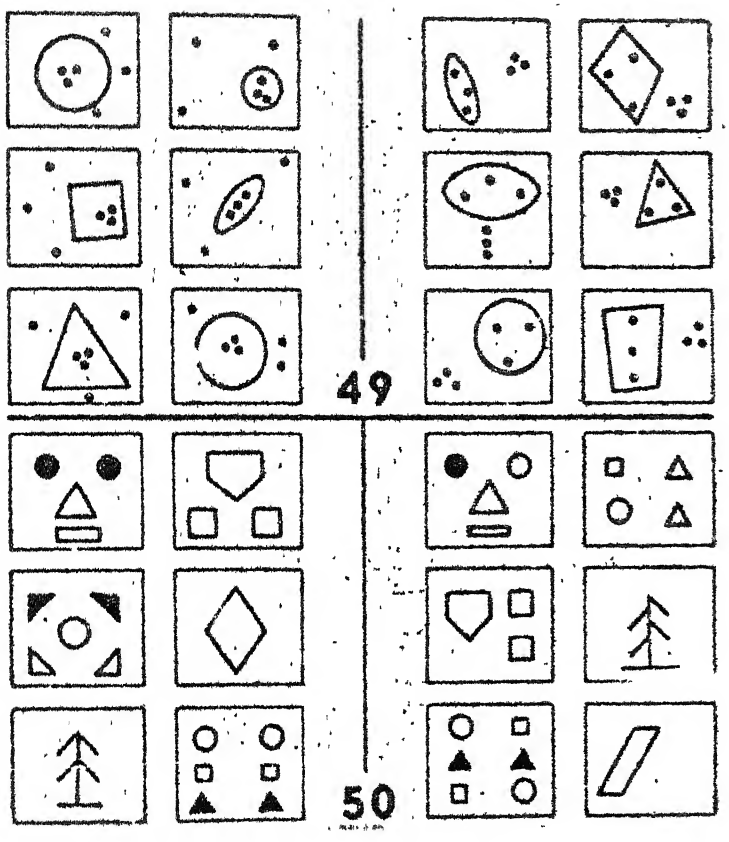

38

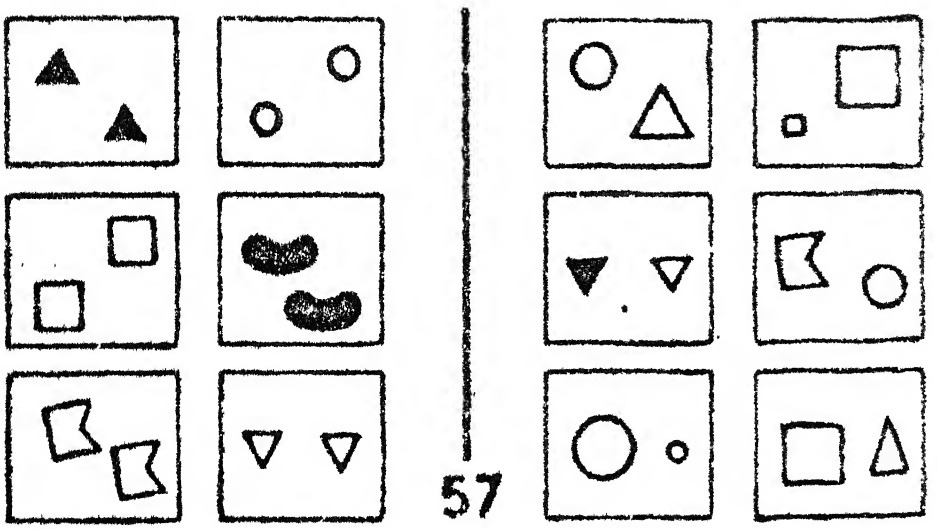


47



48


49
50


57

APPENDIX 3

IMPLEMENTATION DETAILS

We describe here a few details about the implementation of our methodology. The program was coded in LISP, the version used being a superset of UCILISP [C.M.U. version '78], running on a TOPS-10 system. The size of the program was about 200 LISP functions.

Structures were implemented as property-lists of LISP atoms. Heuristics were coded as lambda expressions. If a heuristic was relevant to a particular task, it was placed on the property list of the atom representing the task name. The number of slots in a typical class structure was initially about 10 and grew to a maximum of 20 at the end of the problem-solving session.

APPENDIX 4

AN APPLICATION OF THE METHODOLOGY TO CELL CLASSIFICATION

We describe here an application of our methodology for solving Bongard problems. The new domain we have chosen concerns the problem of cell classification. This example has been taken from [MICHALSKI]. Some reasons for this choice are:

(i) The example vividly illustrates most of the claims we made in chapter 6 regarding the expandability of our methodology. Specifically

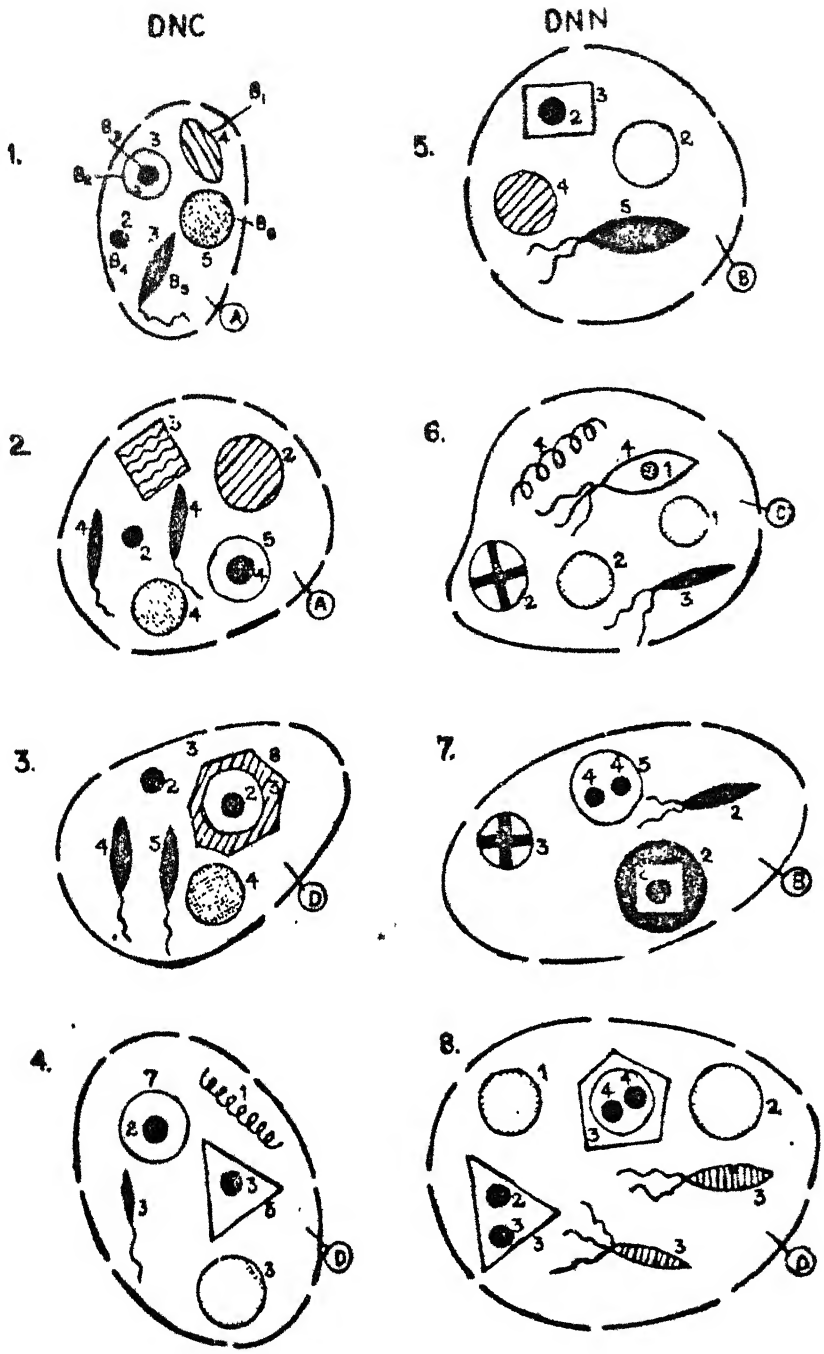
(a) The domain independent nature of the heuristics.

(b) The addition of new descriptors to represent figures and boxes.

(ii) It should prove interesting to compare our approach to this example and the one outlined in [MICHALSKI]. Specifically there the knowledge base is expressed as a set of predicate calculus assertions and a set of inference rules for guiding the rule discovery process.

The cell classification problem is illustrated in figure A4.1. The basic problem is to determine a rule distinguishing the DNC cells from the DNN cells, where the former type are cancerous and the latter-type are non-cancerous in nature.

FIGURE A4.1



The representation chosen for this problem is described below:

1. The set of features and aspects:

	FEATURE		ASPECTS
i.	Number-of-cell-segments	:	{1...10}
ii.	type-of-protoplasm	:	{A,B,C,D}
iii.	shape	:	{triangle, circle, ellipse, heptagon, square, boat, spring}
iv.	texture	:	{blank, shaded, crossed, wavy, solid-black, solid-grey, stripes}
v.	weight (of a cell body)	:	{1...5}
vi.	orientation	:	{N,NE,E,SE,S,SW,W,NW}
vii.	number-of-tails	:	{1,2,3}

As earlier, each feature is composed of one or more aspects. One additional qualification here is that feature 1 and 2 are used to describe the cell itself, while the others are used to describe cell bodies enclosed in the cell.

2. Two basic operations on feature pairs are conjunction and disjunction. So the connectives set C = {and, or}.

3. Spatial-Relations : We include the relation inside-of as a spatial relation relevant to this problem.

4. The set F can be defined as in chapter 2.

Using the above descriptors, figure A4.2 depicts a structure representing the DNC cell 1. The technique used to represent features and their aspects as well as spatial relations is the same as the one we outlined in chapters 3 and 4. The only additional term here is the occurence of features used to describe the 'global' properties of the cell like the number of segments in the circumference of the cell.

Similarly, figure A4.3 depicts a portion of the structure representing the DNC class of cells. The advantages accruing from such a 'structured' representation is clear from these two figures. Since all information about the contents of the DNC class is organised into slots representing features, sub-slots representing their aspects and ternary slots representing cells in which examples of these aspects are to be found, the process of computing a rule distinguishing the DNC class from the DNN becomes quite straightforward.

For example, consider the following rule distinguishing DNC cells from that of DNN cells:

1. Every DNC cell has at least one body with number-of tails = 1.

Figure A4.2

Structure DNC1

is-a : cell

number-of-cell-segments : 8

type-of-protoplasm : A

member : DNC

elements : (DNC1B1 DNC1B2 DNC1B3 DNC1B4
DNC1B5 DNC1B6)

shape :

ellipse : DNC1B1

circle : (DNC1B2 DNC1B3 DNC1B4 DNC1B6)

boat : DNC1B5

texture :

stripes : DNC1B1

blank : DNC1B2

shaded : DNC1B6

solid-black : (DNC1B3 DNC1B4 DNC1B5)

orientation :

NW : DNC1B1

NE : DNC1B5

inside-of : (DNC1B3 DNC1B2)

Note : DNC1 represents cell 1 in the DNC class.

DNC1B1 represents the cell body B1 in the above cell.

Figure A4.3

```
Structure DNC
is-a : class
elements        :  (DNC1 DNC2 DNC3  DNC4)
number-of-tails:
              1 :
                    DNC1 : DNC1B5
                    DNC2 : (DNC2B3 DNC2B5)
                    DNC3 : (DNC3B6 DNC3B7)
                    DNC4 : DNC4B3
 number-of-cell-segments :
                       6   :  DNC3
                       8   : (DNC1 DNC4)
                      10   :  DNC2
 shape :
   Triangle :   DNC4 : DNC4B5
   circle   :   DNC1 : (DNC1B2 DNC1B3 DNC1B4)
                DNC2 : (DNC2B2 DNC2B4 DNC2B6 DNC2B7)
                DNC3 : (DNC3B1 DNC3B3 DNC3B4 DNC3B5)
                DNC4 : (DNC4B2 DNC4B3 DNC4B6 DNC4B7)
   ellipse  :   DNC1 : DNC1B1
                          .
                          .
                          .
                        other slots
                          .
                          .
                          .
```

This rule can be discovered by a process exactly analogous to Bongard problems 21, 24 and 25 (see appendix 2). Consider heuristic 2 (see appendix 1 or chapter 4), reproduced below.

'For a feature f, aspect a and class c
if every figure in c is not described by f, and a occurs in every box in c,
then conjecture that every box in c has a figure described by aspect a of f.'

For f = number-of-tails, a = 1, the above heuristic rule yields a conjucture which is added to the agenda. When this conjecture is checked on class DNN, it fails, and hence yields the rule defining the class of DNC cells.

We list below some other classificatory rules for the DNC cells. It should be evident how such rules could be discovered by existing or new heuristics.

2. Every DNC cell has exactly one body with shaded texture and weight ≥ 3.

3. Every DNC cell has an even number of cell segments.

4. Every DNC cell has at least one boat shaped body with axis of orientation N or NE.

We conclude this brief analysis of the example of an application of our methodology. We have thus demonstrated the domain independent nature of our heuristics and the expandibility of our representation scheme.